

AD-A042 300

NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF
A SIMULATION MODEL FOR THE STUDY OF JOB SCHEDULING POLICY.(U)
JUN 77 E FIEGL

F/G 9/2

UNCLASSIFIED

NL

1 of 2
ADA042300



AD A 042300

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

A Simulation Model for the
Study of Job Scheduling Policy

by

Erik Fiegl

June 1977

Thesis Advisor:

N. F. Schneidewind

Approved for public release; distribution unlimited.

AD No. _____
DDC FILE COPY;



REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Simulation Model for the Study of Job Scheduling Policy.		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis, June 1977
7. AUTHOR(s) Erik/Fiegl		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1977
		13. NUMBER OF PAGES 183
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Simulation Job Scheduling		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A job scheduling simulation model was produced for the purpose of developing an effective Initiator usage policy. While the first part of the study gives a detailed overview of the structure of the OS/MVT Job Management routines the parts thereafter are devoted to describing the simulation model and to an analysis of Naval Postgraduate School Computer		

251450

LB

Center operational data. A user's manual, a demonstration run with results, and a program listing of the simulation model are included.

SECTION 10	
WTS	Write Section <input checked="" type="checkbox"/>
D.C.	But Section <input type="checkbox"/>
UNANNOUNCED JUSTIFICATION	
DISTRIBUTION, AVAILABILITY CODES	
Dist.	AVAIL. AND SPECIAL
A	

DD Form 1473
1 Jan 73
S/N 0102-014-6601

Approved for public release; distribution unlimited

A SIMULATION MODEL
FOR THE STUDY OF JOB SCHEDULING POLICY

by

Erik Fiegl
Kapitaenleutnant, Federal German Navy

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the
NAVAL POSTGRADUATE SCHOOL
June 1977

Author:

Erik Fiegl

Approved by:

W. F. Schneiders

Thesis Advisor

Ronald J. Palmer

Second Reader

W. L. May
Chairman, Department of Computer Science

D. A. Spradley
Dean of Information and Policy Sciences

ABSTRACT

(cont. p. 10)

A job scheduling simulation model was produced for the purpose of developing an effective Initiator usage policy. While the first part of the study gives a detailed overview of the structure of the ^{IBM} OS/MVT Job Management routines, the parts thereafter are devoted to describing the simulation model and to an analysis of Naval Postgraduate School Computer Center operational data. A user's manual, a demonstration run with results, and a program listing of the simulation model are included.

A

TABLE OF CONTENTS

I.	INTRODUCTION.....	10
II.	IBM OS/MVT.....	12
	A. SYSTEM OVERVIEW.....	12
	B. JOB MANAGEMENT ROUTINES.....	16
	1. Reader/Interpreter.....	18
	a. Functional Description.....	18
	b. Input Job Stream.....	20
	c. Input Queues.....	21
	2. Initiator/Terminator.....	23
	a. Functional Description.....	23
	b. Job Selection.....	25
	c. Region Management.....	27
	d. I/O Device Allocation.....	29
	e. Task Attaching.....	30
	f. Step and Job Termination.....	31
	3. Output Writer.....	32
	a. Functional Description.....	32
	b. System Output and Output Classes.....	33
	c. Direct System Output.....	33
	C. REMARKS.....	34
III.	SIMULATION MODEL.....	36
	A. OVERVIEW.....	36
	B. SUPERVISOR MODULE.....	39
	C. READER MODULE.....	40
	D. INITIATOR MODULE.....	41
	1. Job Selection and Waiting for Work.....	41
	2. Region Management.....	42
	3. Device Allocation.....	42
	4. Data Set Allocation.....	44
	5. Direct Access Space Allocation.....	45

6. Step and Job Termination.....	45
E. WRITER MODULE.....	46
F. STATISTICAL MODULE.....	46
IV. DATA ANALYSIS.....	47
A. SOURCE OF DATA.....	47
B. JOB STREAM CHARACTERISTICS.....	51
C. OPERATOR RESONSE TIMES.....	77
D. SYSTEM PARAMETERS.....	80
V. VALIDATION.....	81
Appendix A: USER'S MANUAL.....	86
Appendix B: DEMONSTRATION RUN.....	103
Appendix C: COMPUTER PROGRAMS.....	129
BIBLIOGRAPHY.....	130
INITIAL DISTRIBUTION LIST.....	182
LIST OF TABLES.....	7
LIST OF FIGURES.....	8

LIST OF TABLES

I	System Hardware at NPS.....	49
II	Job Class Definitions at NPS.....	50
III	Distribution of Job Arrivals.....	54
IV	Distribution of Job Steps per Class.....	56
V	Distribution of Input Cards per Job (1).....	59
VI	Distribution of Input Cards per Job (2).....	60
VII	Distribution of Core Used per Step (1).....	64
VIII	Distribution of Core Used per Step (2).....	65
IX	Distribution of Elapsed Step Run Time (1).....	68
X	Distribution of Elapsed Step Run Time (2).....	69
XI	Distribution of Tapes and Disks per Job Class.....	76
XII	Distribution of Operator Response Times.....	78
XIII	Distribution of Job Classes (Validation Runs).....	82
XIV	Initiator Usage (Validation Runs).....	82
XV	Validation Results.....	83

LIST OF FIGURES

1. Structure of OS/MVT.....	13
2. Job Management: Data Flow.....	17
3. Reader/Interpreter: Data Flow.....	19
4. Input Job Stream.....	21
5. Initiator/Terminator: Flow of Control.....	24
6. Job Selection.....	26
7. Utilization of Main Storage.....	28
8. Structure of the Simulation Model.....	38
9. Job Arrivals (April 1976).....	52
10. Job Arrivals (May and August 1976).....	53
11. Job Class Distribution.....	56
12. Histogram: Job Steps per Class (1).....	57
13. Histogram: Job Steps per Class (2).....	58
14. Histogram: Input Cards per Job (1).....	61
15. Histogram: Input Cards per Job (2).....	62
16. Histogram: Input Cards per Job (3).....	63
17. Histogram: Core Used per Step (1).....	66
18. Histogram: Core Used per Step (2).....	67
19. Histogram: Elapsed Step Run Time (1).....	70
20. Histogram: Elapsed Step Run Time (2).....	71

21.	Histogram: Elapsed Step Run Time (3)	72
22.	Histogram: Elapsed Step Run Time (4)	73
23.	Histogram: Elapsed Step Run Time (5)	74
24.	Histogram: Elapsed Step Run Time (6)	75
25.	Histogram: Operator Response Times.....	79

I. INTRODUCTION

The term 'job scheduling' describes the process of deciding the order in which a set of jobs is to be carried out. Within the data processing literature this term is used in three different ways.

In early computer systems, which were controlled by a batch monitor, there was only one job at a time in the computer. The operator had to decide which job would run next. In this connection the term 'job scheduling' was used to describe the manual operation of preparing, selecting, and feeding jobs into the system.

The author uses the term 'job scheduling' in the context of automated

With the development of spooling techniques and multi-programming systems, the manual scheduling operation was automated. Now jobs are fed into the system and spooled. Special job scheduling routines, which are part of the operating system, determine which of the jobs in the queue can be started next. The decision is based upon criteria such as availability of resources and job priority. This kind of 'job scheduling' is sometimes referred to as 'high-level scheduling'. *→ (cont on p 4)*

Once a job is started in a multi-programming or a time-sharing system, it is run concurrently with other jobs. The determination of which job gets the CPU next is the third area where the term 'job scheduling' or 'low-level scheduling' is used.

For this thesis the second definition of 'job scheduling' is used.

The IBM operating system OS/MVT was studied for the following reasons:

- 1) This system is used in the W. R. Church Computer Center at the Naval Postgraduate School and detailed literature about its structure was available.

- 2) The IBM OS/MVT is a general purpose operating system designed to handle a wide variety of applications. Its Job Management routines cover most of the aspects connected with job scheduling.

- 3) In addition, OS/MVT has one interesting feature: Although it is highly automated, certain job scheduling functions (starting Initiators and assigning job classes) still require operator interaction. Immediately some questions arise: Is there an optimal Initiator strategy? Is it possible to support or even replace the manual functions by an automated process?

While the first part of this thesis gives a detailed overview of the structure of the OS/MVT Job Management routines, the parts thereafter are devoted to describing a simulation model and to an analysis of operational data at NPS. The model and the evaluated data could be used to test different Initiator policies.

II. IBM OS/MVT

A. SYSTEM OVERVIEW

The IBM System/360 Operating System (OS) is a general purpose operating system which exists in three different versions: the Primary Control Program (PCP), Multiprogramming with a Fixed Number of Tasks (MFT), and Multiprogramming with a Variable Number of Tasks (MVT).

OS/PCP is intended for small systems and provides only basic functions and sequential job scheduling.

In addition to the basic functions, OS/MFT supports multiple input readers and output writers as well as graphics and telecommunication devices. It also has a time slicing feature and allows concurrent execution of up to 15 programs in fixed partitions.

OS/MVT, which is of special interest for this thesis, includes all facilities of OS/MFT but allows the partition (region) size to vary dynamically depending upon the needs of the particular program. In addition, it has subtasking, roll-in/roll-out, multiprocessing capabilities and a time-sharing option (TSO). Both OS/MFT and OS/MVT provide priority scheduling where programs are multiprogrammed within each priority class. A structural overview is given in Figure 1.

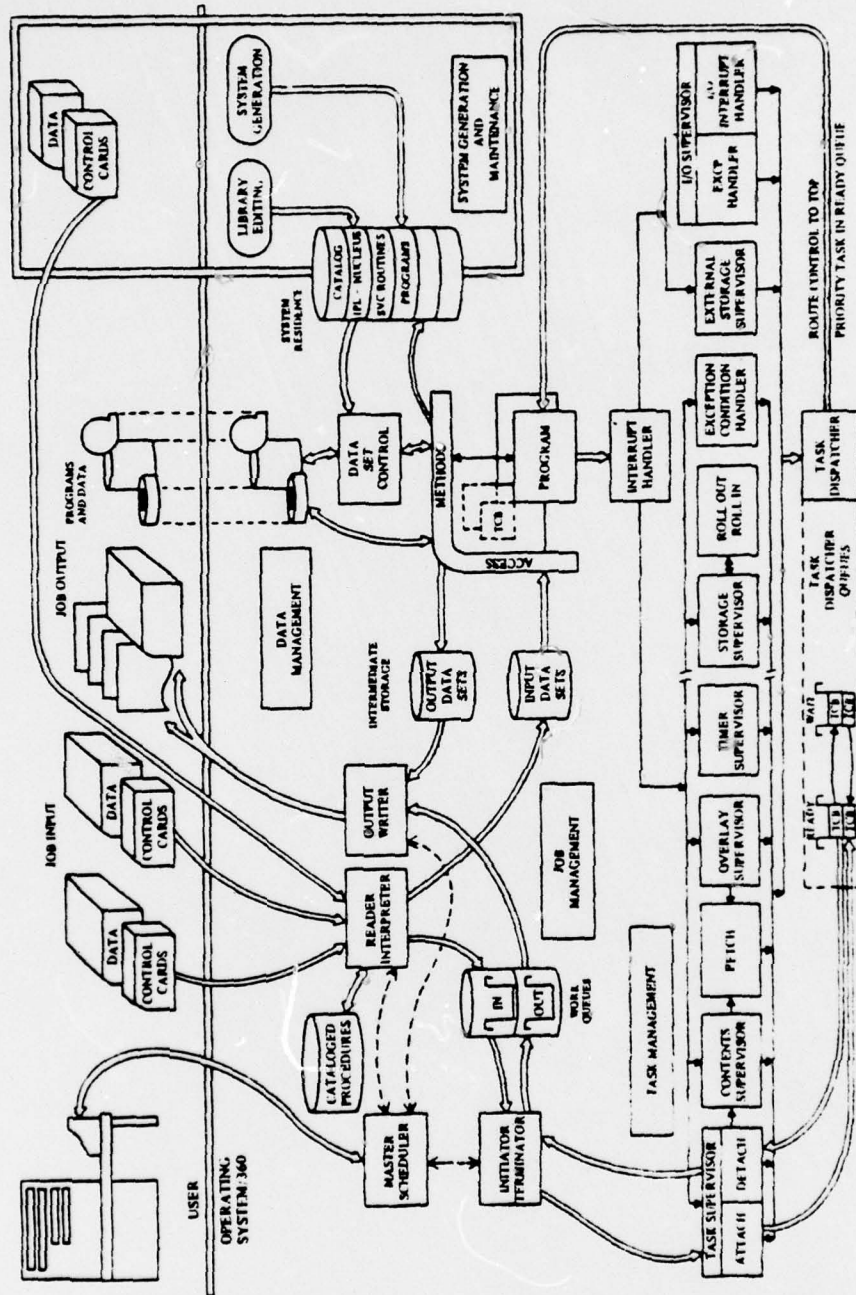


Figure 1 - STRUCTURE OF OS/MVT

Source: IBM OS/360 Chart

OS/MVT consists of two main parts: a Control Program and a set of Processing Programs.

The Control Program is the heart of the operating system. It is designed to manage the overall operation of the computing system and to allocate system resources in order to satisfy user requirements as well as system needs. It consists of several routines which can be grouped into five classes: Task Management, Job Management, Data Management, Volume Management, and Recovery Management.

Task Management routines, often referred as the Supervisor, control the execution of all work done in the computing system. They also serve as an interface between hardware and software. In general, the following functions are performed:

- * handling interrupts
- * supervising tasks
- * controlling programs in main storage
- * controlling main storage itself
- * supervising the timer
- * maintaining the system log

Job Management routines serve as a communication interface between the Control Program on one side and the operator and the users on the other. This communication processing is divided into the following functions:

- * reading, scheduling, and executing operator commands
- * reading the input job stream
- * analysing the Job Control Language
- * initiating jobs for execution
- * obtaining system resources
- * processing the termination of jobs
- * writing system messages and system output

Data Management routines handle the interface between programs and auxiliary storage. This includes:

- * performing data access functions
- * performing input/output support functions
(OPEN, CLOSE)
- * managing input/output buffers
- * maintaining the data set catalog
- * supplying program library facilities

Volume Management routines are used to check the condition of tapes and tape drives. They monitor the number of read and write errors for a given volume and provide error statistics and analysis. For example, a rapidly rising rate of errors would indicate the probability of a deteriorating tape and actions to rescue its contents could be taken.

Recovery Management routines try to recover from CPU and I/O hardware failures and record critical machine and program data in case of machine malfunctions.

Processing Programs , which normally run under the supervision of the Control Program, fall into three categories:

- * language processors (ASSEMBLER, FORTRAN Compiler
PL/1 Compiler, etc.)
- * service programs (Linkage Editor, Loader, System
Utilities, etc.)
- * user programs

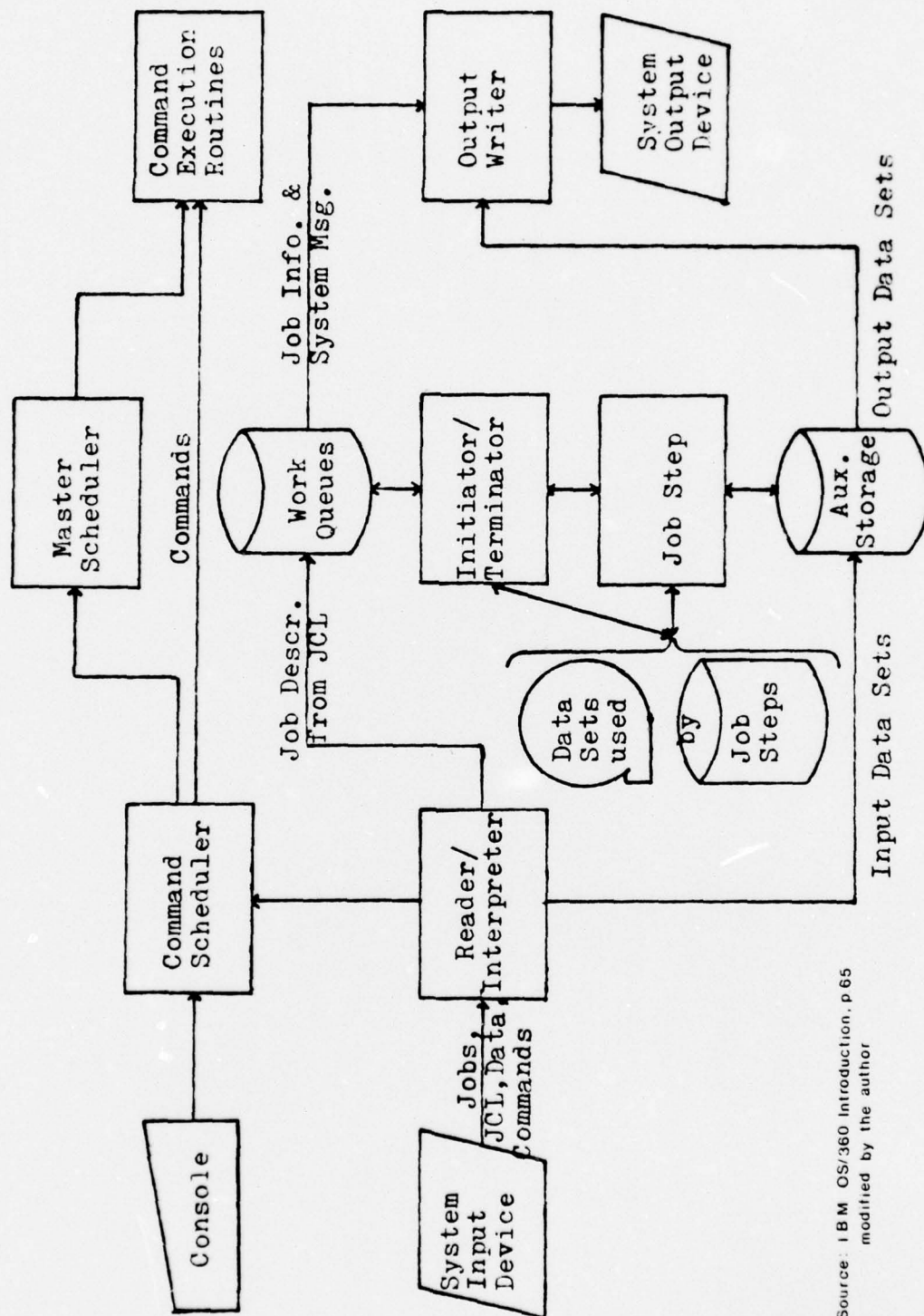
B. JOB MANAGEMENT ROUTINES

Job Management routines can be separated into two categories: command processing and job processing routines.

The command processing routines are the first system tasks which are established during Initial Program Load. They provide an interface between the system and the operator by handling operator commands and writing out system messages to the operator.

The job processing routines handle the input of user jobs, their scheduling, and their termination and exit from the system. This involves three kinds of tasks: Reader/Interpreter, Initiator/Terminator, and Output Writer. All these tasks are created by the operator with certain commands, such as START INIT.id,,,class . The operator may also modify or delete some of these tasks later. Because OS/MVT is a multiprogramming operating system all Job Management routines are executed separately from and concurrently with each other and other system or user tasks.

An overview of the data flow within the Job Management routines is given in Figure 2.



Source: IBM OS/360 Introduction, p 65
modified by the author

Figure 2 - JOB MANAGEMENT: DATA FLOW

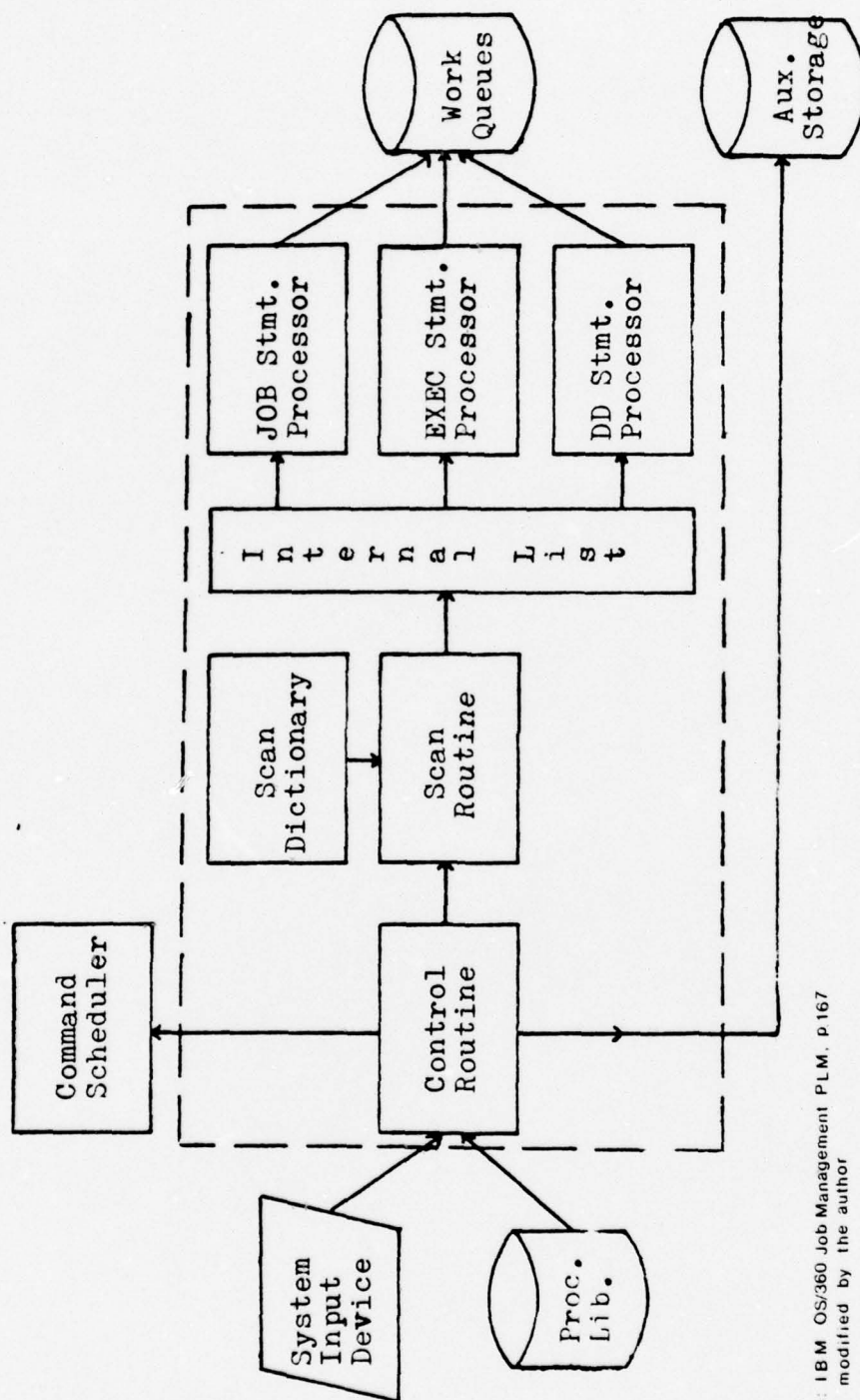
1. Reader/Interpreter

a. Functional Description

The Reader/Interpreter (Figure 3) is responsible for handling the input job stream. It reads it into the system and prepares it for further processing by other system tasks. More specifically, it performs the following functions:

- * reads input job stream and Procedure Library
- * scans and interprets JCL statements and builds appropriate tables
- * creates output queue entries for output data sets
- * places system messages to the user into the output queue
- * writes input data to auxiliary storage and places the appropriate pointer to it into the job input queue entry
- * enqueues the job input queue entry according to job class and priority
- * passes operator commands to the command scheduler

Described above is a combined Reader/Interpreter which operates as a single task. Depending upon the needs of a given installation it is possible to split the functions into separate reading and interpreting tasks. The Reader/Interpreter is started by the operator via a START Reader command. Since more than one input device is allowed in an OS/MVT system it is also possible to start more than one Reader/Interpreter. The performance of the Reader/Interpreter is terminated either when the operator issues a STOP command or when the associated input device signals that the input job stream is exhausted.



Source: IBM OS/360 Job Management PLM, p 167
modified by the author

Figure 3 - READER/INTERPRETER: DATA FLOW

b. Input Job Stream

User jobs are given to the system in the form of an input job stream. This consists of Job Control Language (JCL), input data, and optional operator commands. A typical input job stream is shown in Figure 4.

The JCL provides a description of the job and its resource requirements. The JOB card informs the system about the job itself, such as job name, priority, job class, accounting information, etc. Each job consists of one or more steps. These are defined by the programmer and arranged in the order in which they should be processed. Each step is identified by an EXEC card which gives information about the program to be executed. This program could be system-supplied, like a compiler or the Loader, or it could be a program previously created by the user. Any data set which is accessed or created by a job step must be defined on a DD card. Necessary information are data set name, device, storage size, etc.

The input data are records which are stored on auxiliary storage and later passed to those programs described in a job step. For example, this could be code to be translated by a compiler as well as some numerical data to be processed by a previously created user program.

Certain operator commands could also be part of the input job stream. They are separated and passed on to the command scheduler for further processing.

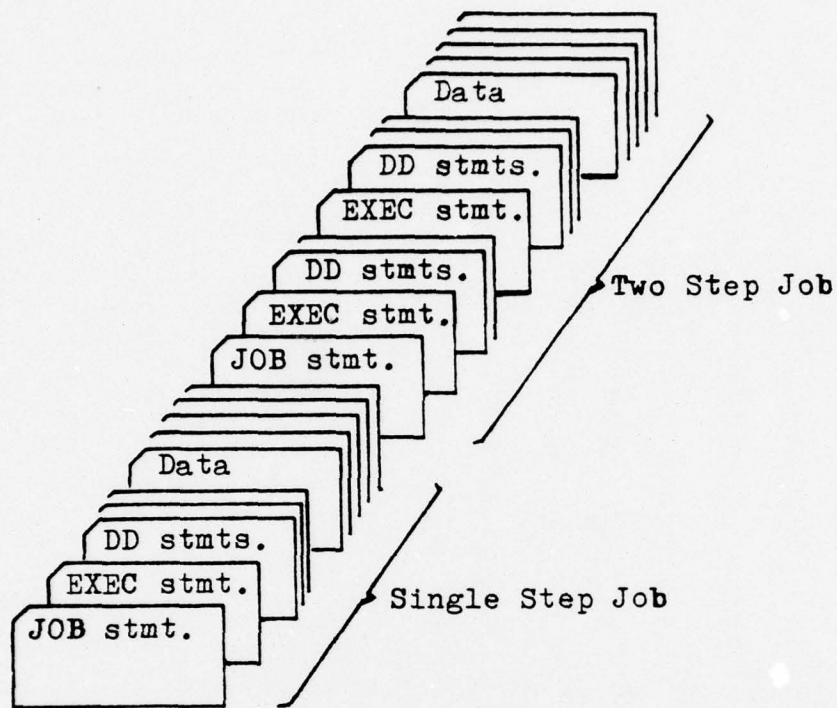


Figure 4 - INPUT JOB STREAM

c. Input Queues

The input queues belong to the work queues. They are temporary storage areas which allow work to be stored according to the input sequence, but to be processed in some user- or operator-defined priority sequence. Not only the Reader/Interpreter, but also the Initiator/Terminator and other system tasks have access to the work queues.

In the OS/MVT there are 76 work queues, 15 of which are input queues and 36 are output queues. They are maintained by a set of routines common to all Job Management tasks.

When the Reader/Interpreter processes the input job stream it translates the JCL into a series of tables [Job Control Table (JCT), Step Control Table (SCT), Step I/O Table (SIOT), and Job File Control Block (JFCB)]. These tables - except those describing an output data set - form an input queue entry. This entry is enqueued when the interpretation of one job is completed. In standard IBM JCL usage, it is placed into that queue defined by the CLASS parameter on the JOB card. Possible classes are A through O. Also under standard JCL usage, the position within the queue is in accordance with the priority specified in the PRTY parameter on the JOB card; for equal priorities it is in the sequence of arrival. If no class and/or priority is specified a default value is assumed by the system, i.e. CLASS=A and PRTY= some value defined at system generation time.

In installations which do not use the CLASS parameter (Naval Postgraduate School) class is determined by user specified resources, e.g. CPU time and core usage, or the appropriate default values. Similarly, for installations which do not use the PRTY parameter (Naval Postgraduate School), position in the queue within class is determined either by a system computed priority or by order of arrival.

2. Initiator/Terminator

a. Functional Description

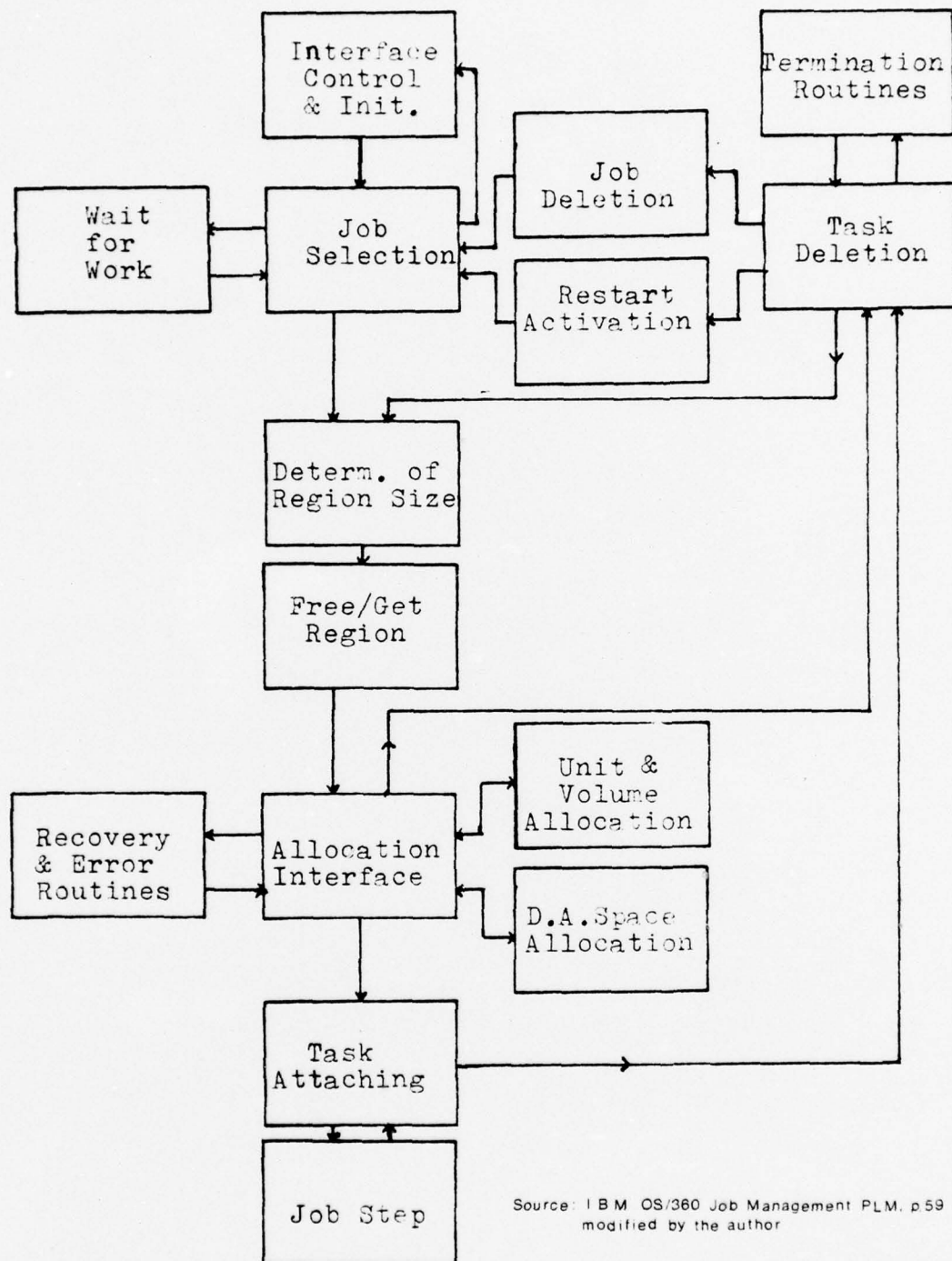
Regardless of how many jobs are read into the system by the Reader/Interpreter no job can be started without the existence of an Initiator/Terminator task. This task is created by the operator with a START command specifying an Initiator and its assigned job classes. An overview of the flow of control within an Initiator/Terminator is given in Figure 5.

The initiating part of the Initiator/Terminator selects jobs and prepares job steps for execution. In particular, the following functions are performed:

- * job selection
- * region management
- * I/O device allocation
- * task attaching

When a job step is complete the termination routines of the Initiator/Terminator release I/O devices and dispose data sets used by this step. They also update appropriate system tables. Upon completion of the last step of a job some additional system bookkeeping is done.

One Initiator/Terminator processes only one job (and within a job only one step) at a time. Multiprogramming is achieved when more than one Initiator/Terminator is started by the operator. Thus, the number of job steps that can be executed concurrently (i.e. the degree of multiprogramming) is equal to the number of active Initiator/Terminators in the system.



Source: IBM OS/360 Job Management PLM, p.59
modified by the author

Figure 5 - INITIATOR/TERMINATOR: FLOW OF CONTROL

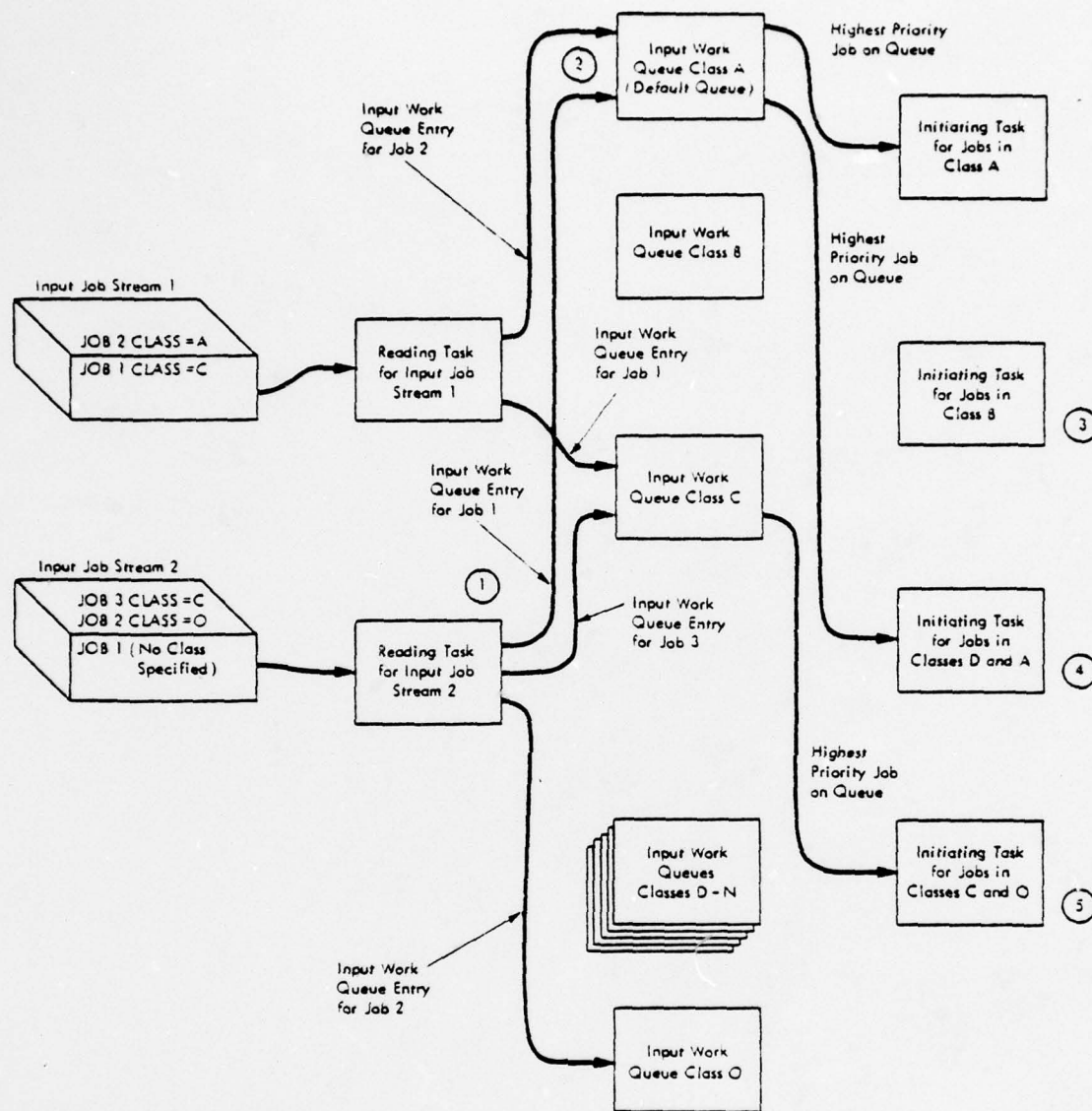
b. Job Selection

When an Initiator/Terminator is started the operand field of the START command contains a list of assigned job classes. The order in which these classes appear determines the order in which the job selection routine searches the job input queues for jobs to be started.

If there are no jobs available in any of the associated classes, then the Initiator/Terminator enters a wait state and sends a message 'Waiting for work' to the operator. This state is kept until an appropriate job arrives or the operator issues a STOP or MODIFY command.

When a job is found the Initiator/Terminator dequeues its entry from the job input queue and passes it to the region management routines for further processing.

Figure 6 gives some examples of Initiators performing job selection or waiting for work. This figure also demonstrates the interaction of Reader/Interpreter and Initiator/Terminator with the input queues.



- 1 The queue for class A is used because no class was specified in the JOB statement.
- 2 Entries in the same input work queue are enqueued according to priority.
- 3 This task is in the wait state if there are not entries in the queue for class B.

- 4 Queue for class D is checked for entries before queue for class A.
- 5 Entries in queue for class C are always processed first regardless of priorities of entries in queue for class O.

Source: IBM OS/360 MVT Guide, p.172

BEST AVAILABLE COPY

Figure 6 - JOB SELECTION

c. Region Management

The region management routines of the Initiator/Terminator determine the requirements for main storage of the current job step. They free the region now being used and request a new region using the FREE MAIN and GET MAIN system macros. The new region size is the larger of either the size required by the job step or the minimum core size needed by the Initiator/Terminator itself. If there is not enough contiguous core in the Dynamic Area the initializing task is put into a wait state until another task releases some core and enough main storage becomes available. When the Initiator/Terminator must wait for work the currently used region is released completely.

Normally the Initiator/Terminator uses the region of the last job step terminated. But at the very first step or at each first step after waiting for work such a region does not exist. For these cases some small routines of the Initiator/Terminator reside permanently in the Link Pack Area. They are capable of requesting a minimum core size in the Dynamic Area to get other Initiator/Terminator routines started.

Space for a region is assigned from the Dynamic Area of the main storage (see Figure 7). The assignment is made in contiguous blocks of 2 K bytes beginning from the highest available address in the Dynamic Area. The smallest region size needed by an Initiator/Terminator is 12 K bytes. To improve system performance a minimum region size of 52 K bytes is recommended by IBM.

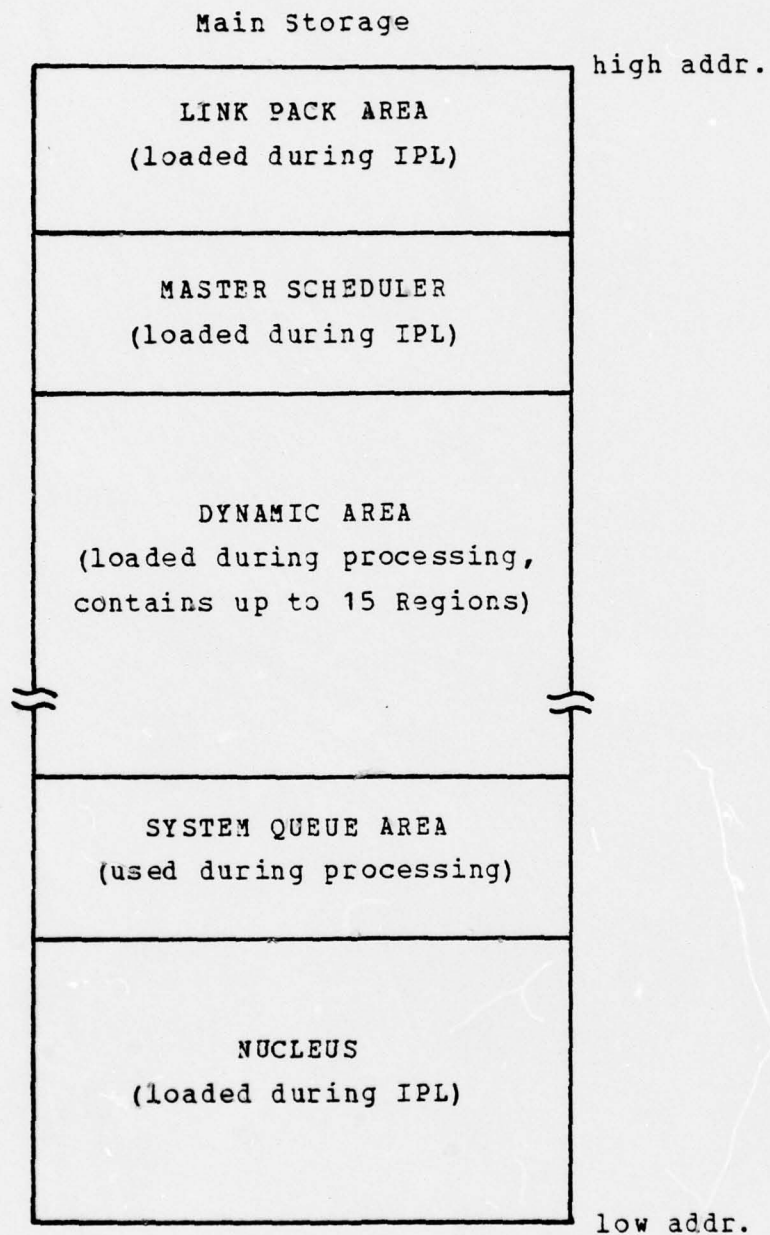


Figure 7 - UTILIZATION OF MAIN STORAGE

d. I/O Device Allocation

The I/O device allocation routines handle the I/O requirements specified in the DD statement for each job step. If some requirements are not specified completely then information gathering routines search catalogs, check status of devices, or use default values to fill these information gaps implicitly.

The input data sets used by a step are located and the device allocation routines determine if any I/O devices are available for these data sets. A step cannot be initiated unless there are enough devices - for both input and output data sets. If there are sufficient devices available they are allocated to that job step. Otherwise a message is issued to the operator. He may put the Initiator/Terminator into a wait state until enough devices are available, or he may cancel the job.

After all devices are allocated the Task Input Output Table (TIOT) is built. It contains pointers and necessary information for other system routines to allow processing of each data set used by the job step.

For volumes which are not yet mounted mount messages are issued to the operator. For output data sets which require direct access space the amount of space is calculated and checked against the available space. If enough space is available it is assigned to that job step, otherwise the allocation recovery routine is entered. If there is another task active using temporary direct access space, then this routine puts the Initiator/Terminator into a wait state until this space is released. Otherwise the recovery routine informs the operator. He may put the

Initiator/Terminator into a wait state until he can make some space available, or he may cancel the job.

Before the I/O device allocation routines can pass control to the task attaching routines, they have to wait until all volumes, for which mounting requests have been issued, are mounted. A final check is made to verify that the mounted volumes are correct. If an incorrect volume has been mounted, a demount instruction followed by a new mount message is issued to the operator. The I/O device allocation routines again have to wait until this error is corrected.

e. Task Attaching

When all resource requirements of a job step can be satisfied the final operation prior to starting this step is to attach the user task to the Supervisor. For this purpose the attaching routines gather information (dispatching priority, remaining job run time, etc.) needed by the Supervisor. This information is placed into a Task Control Block (TCB) and queued into the TCB queue.

The initiation of the job step is now complete and the step will run under the control of the Supervisor. The Initiator/Terminator task is placed into the wait state until this step is to be terminated.

f. Step and Job Termination

A job step is terminated either normally when it is complete or abnormally when an error prevents further processing, when a specified time interval has expired, or when the job is cancelled by the operator. In any case the Supervisor activates that Initiator/Terminator which started this job step. The termination process is then done by this Initiator/Terminator using the region of the just ended step.

The termination routines direct the disposition of data sets and the release of I/O devices used by the job step. They also update appropriate system tables.

If there are more steps control is passed to the region management routines to initiate the next step.

If the last step of a job is terminated some additional processing must be done. The job entry from the input queue is removed completely and entries of the job's system output data sets are enqueued into the appropriate output work queues. Control is then passed to the job selection routines of the Initiator/Terminator.

3. Output Writer

a. Functional Description

In a multiprogramming system a job is usually not allowed to use a printer or a punch directly, even if it is the only job in the system at that time. Hence, a job generally writes on intermediate output data sets on direct access devices. When a job terminates, pointers to those data sets as well as system messages concerning that job are enqueued into output queues. Such messages and data sets are then processed by a System Output Writer.

An Output Writer is created by the operator as a result of a START Writer command. One parameter of the START command specifies the associated output device (printer, punch, or tape), another parameter describes a single output class or a group of up to eight classes. More than one Output Writer may be started, depending upon the needs of an installation.

An Output Writer controls the writing of all system output within its specified class(es). It enqueues entries from the output queues and performs the required output operation on its assigned system output device. When all entries of the assigned class(es) have been processed, the Output Writer is placed into a 'wait' state. It is again made ready when a job terminates and the Initiator/Terminator places an entry into a queue associated with this Output Writer.

b. System Output and Output Classes

The system output consists of messages from the operating system to the programmer and of data sets created by the job and designated by the programmer in a DD statement. The messages and pointers to the data sets are placed as entries into the system output queues by the Reader/Interpreter and by the Initiator/Terminator.

The system output is divided into 36 classes and there is one output queue corresponding to each class. The classes are named with single letters (A-Z) or digits (0-9). The names have no inherent meaning but are simply used to group output of similar characteristics. There might be, for example, one class for output to a printer, another class might be for punched output, and a third class might be for output written on tape for later printing.

c. Direct System Output

By use of Direct System Output Writers there is a possibility that jobs can directly use system output devices such as printer, punch, or tape. Direct System Output Writers are started by the operator and assigned to a certain device and to specific output classes. The assignment to a job is made by the I/O device allocation routines of an Initiator/Terminator. The selected writer is then tied to that job for the duration of all job steps. When the job writes its output, the output will go directly to the specified device. In addition, this job may also produce output of other output classes which will be spooled and later processed by normal Output Writers as described earlier.

C. REMARKS

As described earlier an Initiator/Terminator serves only one job at a time. This means that, once the Initiator/Terminator routines have selected a job, they are tied to it until this job has terminated. This might be a good approach, since it establishes a well defined relationship between an Initiator/Terminator and a user's job. However, there is an obvious disadvantage which shows up whenever an Initiator/Terminator has to wait for a requested resource. When waiting for data sets to be mounted or for other operator interactions, this waiting time could extend from a few seconds to several minutes. During that time the Initiator/Terminator remains dormant. Since it is tied to one job only it cannot serve any other job. This means that during waiting periods the degree of multiprogramming is decreased by one.

OS/MVT allows and even requires a high amount of operator interaction. At Initial Program Load the operator has to set system parameters and he must start and assign classes to Readers, Writers, and Initiators. Later he may stop, modify or start new system tasks, or he can hold, reset, or cancel certain jobs and can connect and disconnect specific devices. According to IBM this provides a great amount of flexibility. At any time the performance of certain system tasks could be tailored to the need of a given job load. The ability to supervise large parts of the operating system allows the operator to handle even very extreme cases and emergency situations which could not be handled by a fully automated system.

The disadvantage of this approach is that the performance of this sophisticated and otherwise highly automated operating system is mainly influenced by the experience and performance of the operator.

The job scheduling algorithm used in OS/MVT requires that all resources be pre-allocated to a job step. That means , no job step can be started until sufficient main memory and all requested data sets and devices are available. The lack of only one of the needed resources will cause - depending upon the circumstances - either a wait until this resource becomes available , or a cancellation of the job step, or an intervention request to the operator.

The purpose for this kind of resource control is to avoid deadlocks which could arise from simultaneous use of the same resources by several tasks. This strategy may be sometimes inefficient and costly since some of the resources allocated to a job step remain unused for a long period of time or they may not even be used at all. Practice has shown that the possibility of a deadlock between several tasks initiated by one or more job steps has not been eliminated completely. There is a possibility that the use of ENQ and DEQ macros to control resource allocation may create a 'circular wait'.

III. SIMULATION MODEL

A. OVERVIEW

The model simulates the main functions of the IBM OS/MVT Job Management routines. In general, the overall structure of the operating system is reflected in the structure of the model, but since OS/MVT is a very complex system some simplifications and limitations are necessary. They are described in the following parts.

The purpose of the model is to test different Initiator strategies under certain job loads and operating conditions. To achieve this goal the number of Initiators (up to 15) and their associated job classes (up to 8 per Initiator) can be varied during the simulation. The model also allows important system parameters (size of main memory, input spool space, number of I/O devices, etc.) to be entered. By varying these parameters the simulation program can be tailored to a certain extent to a given environment.

The job stream used during the simulation is generated by a job generating module. This module can be modified to allow generation of job streams with different characteristics. Some statistical routines collect and print statistical and performance data upon user request.

A special problem is the simulation of the time used by Job Management routines, by other system tasks, and by the different user jobs. The basic time measurement in the model is elapsed step run time. This is the wall clock time counted from the beginning of step initiation to the end of step termination. The elapsed job run time is the sum of all elapsed step run times of a job. Included in this time is the CPU time used by the job, the time waiting for I/O , as well as the time used by the Job Management routines and other system tasks. Since the elapsed step run time has a range of one second to several minutes, one second is used as the basic time unit in the simulation model.

The programming language PL/1 was chosen for several reasons:

- * it is a block-structured language
- * it allows good data structuring
- * it is easy to use for I/O routines
- * it is well supported at the Naval Postgraduate School, where the model was developed

In addition, PL/1 allows nearly unrestricted variable names. This makes the program more readable and self-documenting.

A functional overview of the simulation model is given in Figure 8.

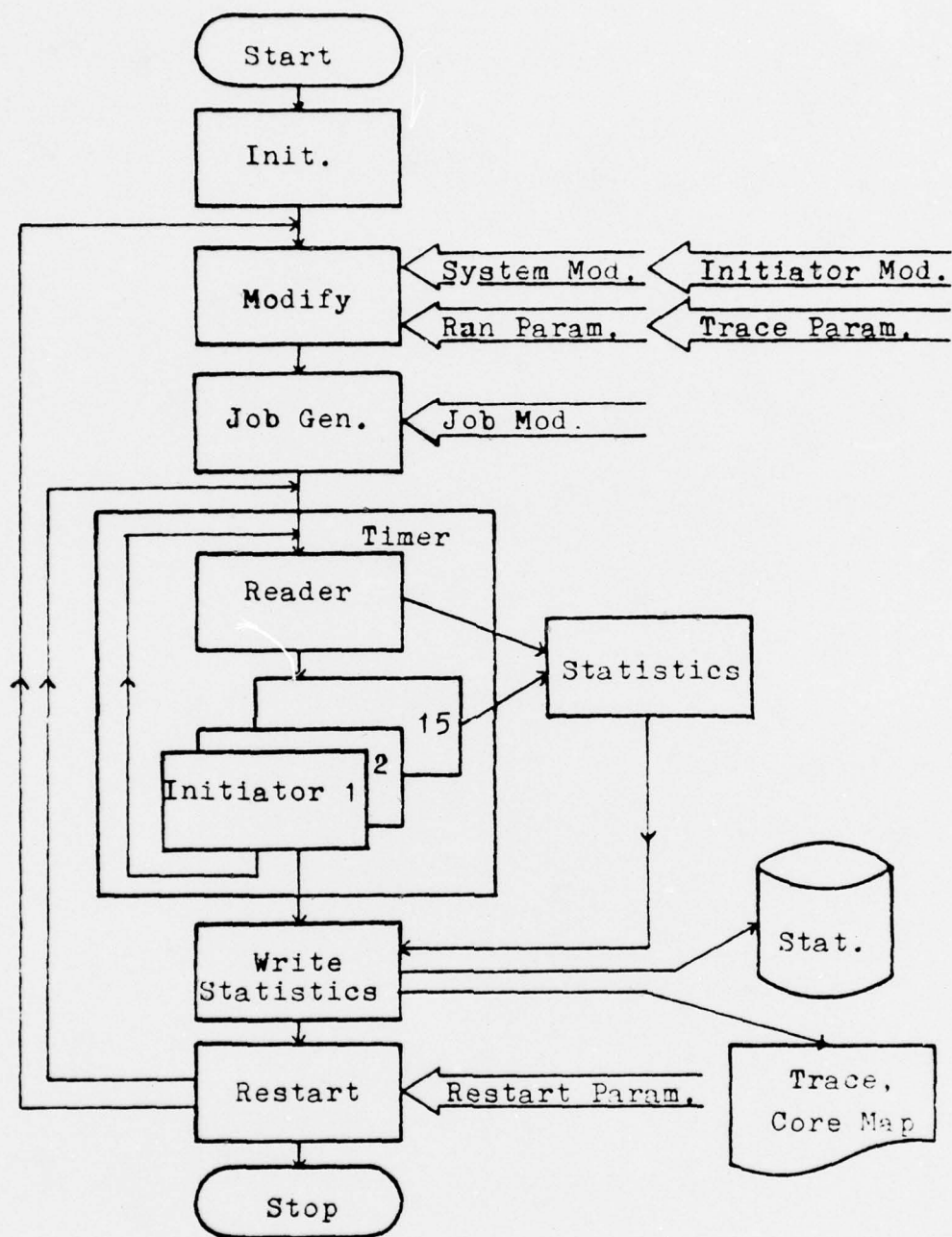


Figure 8 - STRUCTURE OF THE SIMULATION MODEL

B. SUPERVISOR MODULE

The supervisor module initializes and drives the entire simulation program. When it calls the initialization and modification routines, the user may enter the following parameters:

- * system modifications:

- main memory (high address)
- main memory (low address)
- number of disk drives
- number of tape drives
- amount of input spool space
- amount of public direct access space

- * run parameters:

- number of jobs to be read
- simulation time
- job stream modifications

- * Initiator modifications:

- number of active initiators (up to 15)
- associated job class(es) for each initiator

- * trace parameters:

- simulation trace
- map of main memory usage
- statistics gathering

After these parameters are entered the timer module gets control. This module checks the simulation time table, which contains the times when the Reader and each active Initiator need attention. The timer always calls the next module, which is responsible for updating the attention time. This process is terminated when the simulation time or the input job stream is exhausted, whichever comes first.

At the end of each simulation step the user has the choice to stop or restart. If restart is chosen he may run the simulation with the same or new parameters.

C. READER MODULE

It is assumed that the Reader is active during all simulation steps and that it resides in the upper part of the dynamic area in main memory. The user must note the amount of core used by the Reader when entering the main memory high address parameter.

During the initialization phase the job generating module places the requested number of jobs and their characteristics into the input job stream and also sets the time of the first job arrival into the simulation time table. When the Reader is called by the timer module it takes the next job from the input stream and enqueues it according to its class and priority into one of the job input queues. Then the Reader determines the time of next job arrival and places this time into the simulation time table as its new attention time.

If the input spool space is exhausted the reading and enqueueing of jobs is delayed until another job terminates and enough spool space becomes available. Since the supervisor ends the simulation run after the requested number of jobs has been read, the Reader will never be called when the job stream has been exhausted.

D. INITIATOR MODULE

The Initiator module simulates the functions of job selection, waiting for work, region management, device allocation, data set allocation, direct access space allocation, step termination, and job termination. All information necessary to perform these functions is maintained in an Initiator table. Since the dimension of this table is 15 it is possible to run 15 Initiators concurrently. Each Initiator updates its time of next attention in the simulation time table.

1. Job Selection and Waiting for Work

An Initiator can be associated with up to 8 different job classes. To find the next job the input queues are searched in the order in which the classes were assigned to Initiators by the user.

If there is no job of the appropriate class in the queues, the Initiator releases its region and is put into a 'wait for work' state. This state is kept until a new job arrives. Then the Initiator gets a new region of a pre-defined minimum size and the queues are searched again. If a job is found it is dequeued and associated with its Initiator for further processing.

2. Region Management

When a job is selected the region size of its first step is determined and the region currently used by the Initiator is released. The new region is allocated from the top of the Dynamic Area in main memory. If insufficient contiguous core is available, the Initiator is placed in a 'wait for core' state. It is activated again for a new region allocation when another job ends and core is released. The region management routines are called at the beginning of each job step.

It is assumed that the size of the Dynamic Area is fixed during the simulation. However, the user must set the upper and lower addresses. He can account for the size of the system queues by setting the appropriate lower address. He must also account for the amount of storage used by system tasks, by Reader(s), Writer(s) and other permanent programs by setting the appropriate upper address.

3. Device Allocation

In general, the allocation of I/O devices and I/O channels is not simulated in the model. Most devices are physically shareable (data cell, disks) or are made shareable using spooling techniques (card reader, printer, plotter). Evaluation of system logs has shown that normally all requests to such devices can be satisfied by the system immediately. The time overhead required for selection, allocation, and spooling is included in the elapsed run time of each job step.

However, this simplification is not valid in case of tape drives and disk drives with removable disk packs. The allocation of these devices sometimes requires operator interaction or causes long additional waiting times until a requested tape or disk drive becomes available.

The device allocation routines handle the tape and disk requests of each job step. If a device is not available, an operator interaction is simulated. The operator answer could be 'cancel' or 'wait'. In the first case the whole job is abended; in the second case the Initiator is put into a 'wait for device' state. Whenever another job terminates the device allocation routines are activated again until all outstanding device requests can be satisfied for the current job step. In order to avoid long waiting times, all jobs which request more devices than are installed in the system are abended.

The type of operator answer ('cancel' or 'wait') and his response time are drawn from a probability distribution.

The number of tape and disk drives can be set by the user, thus tailoring the simulation model to his needs.

4. Data Set Allocation

Only those data set allocations are of interest which require operator interaction, thus causing additional waiting time. It is assumed that for every requested tape and disk drive an appropriate volume has to be mounted. By placing the Initiator into a 'wait' state, the data set allocation routines simulate the time needed by the operator to perform the mounting.

Independant of mounting requests are verification requests. Some data sets require an operator response to verify that a user is authorized to access a data set. This case is also simulated by the data set allocation routines. Since, for the model, the operator response time is of greater interest than the reason for an operator interaction, this case could also be used to account for any additional operator request which is otherwise not covered (channel separation request, etc.).

The response time for mounting disks, mounting tapes, or answering other requests is drawn from a probability distribution as described earlier for the device allocation routines. Also the possibility of job cancellation is included in the model.

5. Direct Access Space Allocation

Only the allocation of temporary space on public direct access devices is simulated. The total amount of public space within the system can be set by the user. If a space request of a job step can not be satisfied the space allocation routine checks if there are other job steps active. If not, the current job will be abended, since its request can never be granted. Otherwise the Initiator is placed into a 'wait' state until another job ends which might release some temporary space on public direct access devices.

6. Step and Job Termination

At the end of a step all requested disks and tapes are released and given back to the system. Temporary space on public direct access devices, however, is kept until job termination. If there is another step to process, control is given to the region management routines to start the next step.

At normal job termination as well as in case of job abending all system resources (tapes, disks, public direct access space, and input spool space) are released. The Initiator table is cleared and a new job can be selected. Job termination is also posted to the Reader which might be waiting for input spool space and to other Initiators which are in a state of waiting for system resources.

E. WRITER MODULE

Spooled system output only is assumed for this simulation. That means that the Writer works independently from and concurrently with the Reader and Initiators. Since the amount of overhead due to multiprogramming with the Writer is already included in the elapsed job run times, no Writer function has to be simulated. As mentioned earlier the user, however, must deduct the core size used by the Writer from the top of the Dynamic Area in main memory.

F. STATISTICAL MODULE

Several statistical routines gather Initiator performance data. These data are maintained in a statistical table which can be written on a file upon user request. This file must then be processed and evaluated by a supporting statistical evaluation program.

As a second choice the user can request a simulation trace. Similar to the logs at the operator's console all important events (job starting, job termination, initiator waiting for work, mount requests, etc.) are printed out. As a third choice a map showing the utilization of main memory at the end of each simulation step can be printed.

Examples of a trace, a core map, and some evaluated performance data are shown in Appendix B.

IV. DATA ANALYSIS

A. SOURCE OF DATA

To drive the simulation model certain information about input job stream characteristics, system configuration, operator response times, etc. was necessary. To gather data four main sources were used:

The first source was the IBM 360/67 computer center at the Naval Postgraduate School. An overview of the hardware configuration is given in Table I; the job class definitions and the priority policy are listed in Table II. With the installation of the HASP spooling system in September/October 1976, the Quickrun class was replaced by the input class 0, which was restricted to jobs using certain cataloged procedures only, using up to 180 K of core and up to 20 seconds of CPU time. These restrictions were nearly the same as for the Quickrun class, but since some changes were made in the cataloged procedures about half of all previous class A jobs together with nearly all Quickrun jobs would now qualify for the new class 0. For the validation runs the characteristics of the Quickrun class were simulated.

The second source was data collected from the System Management Facility (SMF) routines. These routines gathered statistics about every job processed by the computer system. Contained therein were: job name, job class, job priority, job arrival time and date, job starting time and date, job

completion code, number of input cards, number of job steps, requested and used core per step, CPU time per step, elapsed time per step, sysout records per step, etc.

Only SMF data of the period from February to August 1976 were usable for the purpose of this study. Before this time period a completely different job class and priority specification was in effect. After this period some parameters used for the simulation model were no longer recorded due to the change to the HASP spooling system. So SMF tapes of April, May, and August 1976, containing data of about 75,000 jobs, were evaluated.

As a third source the complete set of system logs of August 1976 was available and used to extract certain parameters. These parameters included the number of Initiators and their associated job classes, the number of other system tasks active at the same time, and upper and lower addresses of the Dynamic Area in main memory.

Since the SMF tapes did not provide information about usage of tapes and disks the system logs were also used to count the number of tape and disk mount requests and to evaluate data such as operator mounting times, operator response times to other system requests, and the number of job cancellations by the operators.

Last, but not least, the operators themselves and other members of the computer center staff at NPS provided valuable input for the collection and evaluation of system parameters.

IBM 360/67 at NPS

NO.	UNIT	DESCRIPTION
2	2067-2	Processing Unit
1	2167-4	Configuration Control Unit
2	1052-7	Console Typewriter
2	2860-2	Selector Channel
2	2870-1	Multiplexor Channel
3	2365-12	Processor Storage (256K Bytes each)
5	MM365-12	Core Storage (256K Bytes each) - Lockheed
1	2820-1	Drum Control
1	2301-1	Drum Storage (4 M Bytes)
3	2841-1	Disk Control
8	2311-1	Disk Drives (7.25 M Bytes each)
1	2314-1	Disk Unit (8 Drives, 29 M Bytes each)
2	5314	Disk Control - Potter
16	4314	Disk Drives - Potter (29 M Bytes each)
1	2321-7	Data Cell (400 M Bytes)
1	3803-1	Tape Control
5	3420	Tape Drives
1	2803-1	Tape Control
2	2402-1	Tape Unit (2 Drives each)
1	2821-1	Control Unit
1	2821-2	Control Unit
2	1403-N1	Printer
1	2501-B2	Card Reader
1	2540-1	Card Reader/Punch
1	110	Plotter Control - CALCOMP
2	765	Plotters - CALCOMP
1	2702-1	Transmission Control Unit (30 Ports)
24	2741	Communication Terminals
8	----	Video Display Units (assorted vendors)
1	2250-1	Graphic Display Unit
1	Tek4012	Display Terminal - Tektronix
1	Tek4610	Hard-Copy Device - Tektronix
1	PIX	Paradyne PIX/Remote Job Entry
1	2701	Data Adapter Unit (with PDA)

Table I - SYSTEM HARDWARE AT NPS

EFFECTIVE 1 FEB. 1976

JOB CLASS DEFINITIONS

CLASS REGION TIME TAPE/JOBSTEP

Q	QUICKRUN		none
A	180K	20s	none
B	180K	2m	≤2
C	250K	5m	≤2
D	250K	5m	>2
E	350K	5m	≤2
F	400K	30m	none
J	>400K	>30m	none
K	>400K	>30m	any

Comments:

1. Execution in each class will be on First-come First-served (FCFS) basis.
2. Classification scheme ignores SYSOUT and SYSDA requirements. Printing priority is considered separate from execution priority and is based on the actual number of lines generated.

Table II - JOB CLASS DEFINITIONS AT NPS

B. JOB STREAM CHARACTERISTICS

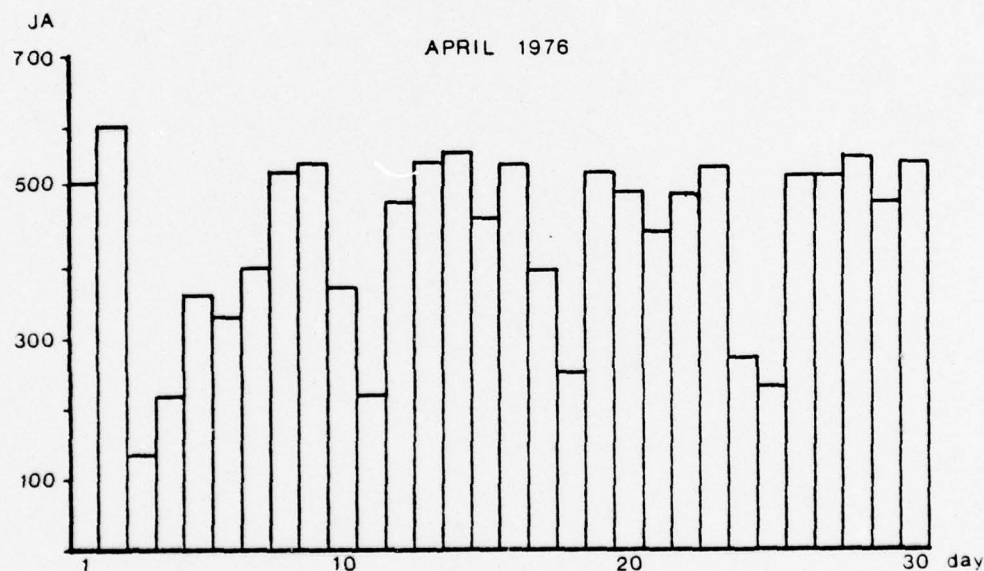
Very important for an effective simulation run were parameters which characterized the input job stream. Members of the computer center staff and students had analyzed the input job stream at the NPS computer center. But since these studies were based on jobs rather than steps, as required by the simulation model, these studies were not usable and a new evaluation had to be made.

Most of the job characteristics were extracted from the SMF tapes. When working with these tapes a few problems arose. There was no class D job observed and the number of J and K class jobs was very small. In addition some jobs in undefined job classes were present. An explanation for this was that the operators used to start one Initiator with an undefined job class. Then they reset jobs from classes J and K and the very few jobs from class D, and selected these manually for initiation. For the simulation model, classes D, J, K, and all undefined classes were collected into one class K.

Also some jobs used more core than allowed by their job class. The explanation again was that operators reset jobs from one class to another. During the evaluation these jobs were filtered out and added to job class K.

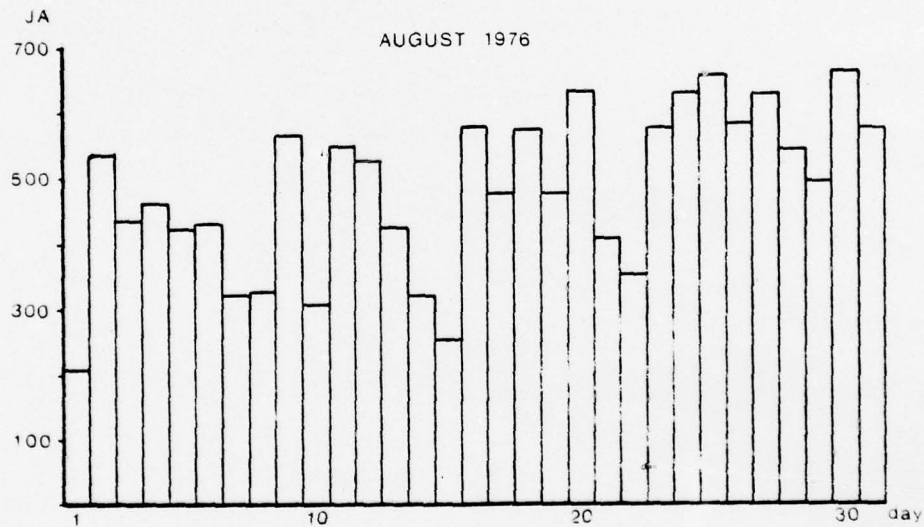
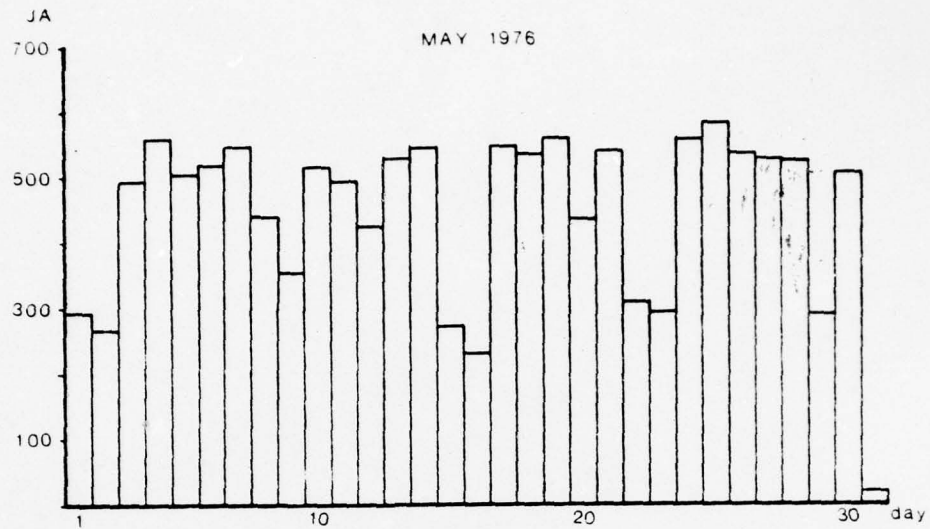
Elapsed time and core used were not recorded for the Quickrun jobs. Since these jobs had the same time and core restrictions as class A jobs (core up to 180 K Bytes, CPU time up to 20 seconds), the class A distribution was assumed.

To obtain relatively stable, but still representative data the time period from 10 a.m. to 5 p.m. each day was selected. The data collection was further restricted to those days with more than 500 job arrivals within this time. Figure 9 and Figure 10 show that 47 days of the months April, May, and August 1976 met these requirements. With this approach untypical conditions which occur at night and on weekends and holidays were eliminated. Although the time of observation covered only about 15% of the total hours within the three month period, 25,532 or more than one third of all job arrivals were included.



JA : Number of job arrivals between 10 a.m. and 5 p.m.

Figure 9 - JOB ARRIVALS (APRIL 1976)



JA : Number of job arrivals between 10 a.m. and 5 p.m.

Figure 10 - JOB ARRIVALS (MAY AND AUGUST 1976)

To obtain the distribution of job arrivals a 2-hour period in each month was selected at random and the arrivals per minute were counted. As shown in Table III in each job class the distribution was very close to a Poisson distribution. In fact, the observed values easily passed a 95 percentile chi-square test to match the theoretical values. Thus for the job arrivals in the simulation model a Poisson distribution with exponentially distributed interarrival times was used. The mean job arrival rate was 1.294 jobs per minute.

	class A		class B		class C	
i	F _o	F _{th}	F _o	F _{th}	F _o	F _{th}
0	67	66.41	88	87.43	105	104.15
1	39	39.29	26	26.69	13	14.75
2	11	11.62	6	4.38	2	1.05
3	2	2.29	0	0.46	0	0.05
4	1	0.34	0	0.04	0	0.00
5	0	0.04	0	0.00	0	0.00

	class E/F		class K		class QR	
i	F _o	F _{th}	F _o	F _{th}	F _o	F _{th}
0	118	118.01	109	109.49	106	105.02
1	2	1.97	11	10.04	12	14.00
2	0	0.02	0	0.46	2	0.93
3	0	0.00	0	0.01	0	0.04

F : observed number of 1-min. intervals with i arrivals
F_o : theoretical number of 1-min. intervals with i arrivals
F_{th} assuming Poisson distribution

Table III - DISTRIBUTION OF JOB ARRIVALS

Other parameters evaluated from the SMF tapes were distribution of job classes, number of steps per job, number of input cards per job, core used per step, and elapsed time per step. The probability distributions are given in Table IV to Table X; histograms are provided in Figure 11 to Figure 24.

It was felt that the distribution of elapsed step time might be approximated by a Gamma or possibly a Weibull distribution. Although a great amount of work was spent to match the observed values with those theoretical functions no relationship could be found.

In the simulation model the amount of public direct access space was one input parameter. The storage of system output records was only one part of this space, but other data were not available. An evaluation of the job completion codes, however, showed that within the observed time periods no job abended because of lack of public direct access space. Thus for the simulation runs no public direct access space was requested.

The number of disk and tape mount requests and the number of other system requests were evaluated from the system logs of August 1976. Only the total number of requests could be counted, but information about the associated job classes was not available. For the simulation model it was assumed that the probability of requests was the same in all job classes which qualified for the appropriate type of requests. The distribution is given in Table XI.

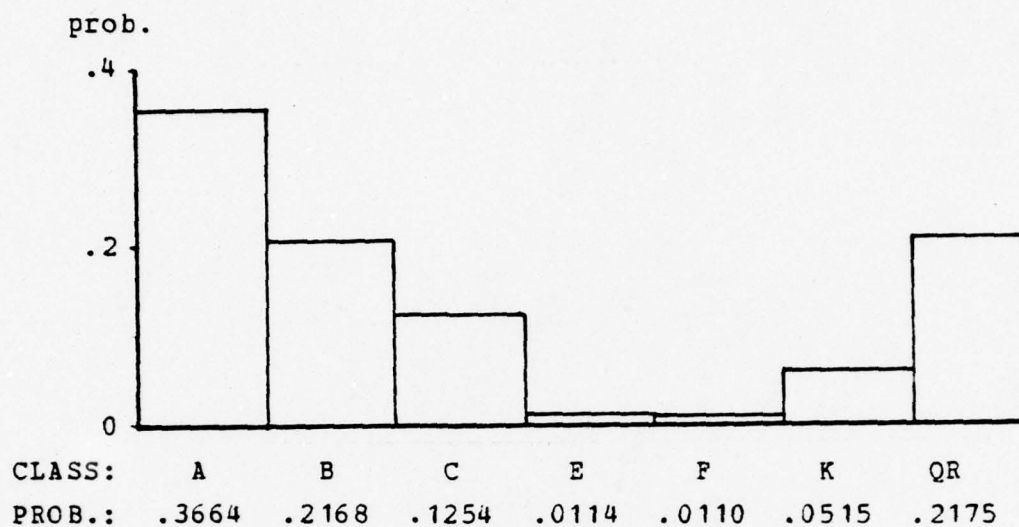


Figure 11 - JOB CLASS DISTRIBUTION

STEPS	A	B	C	E	F	K	QR
1	.4180	.3226	.5420	.3655	.1352	.3934	.5524
2	.0531	.0878	.0615	.1586	.0569	.0845	.4476
3	.5069	.4767	.2877	.4483	.6121	.4155	.0000
4	.0086	.0367	.0285	.0103	.0107	.0350	.0000
5	.0123	.0517	.0619	.0103	.1530	.0471	.0000
6	.0006	.0216	.0172	.0035	.0249	.0152	.0000
7	.0000	.0025	.0006	.0000	.0000	.0046	.0000
8	.0000	.0002	.0000	.0000	.0036	.0008	.0000
9	.0003	.0000	.0000	.0035	.0036	.0015	.0000
10	.0001	.0002	.0003	.0000	.0000	.0000	.0000
11	.0001	.0000	.0003	.0000	.0000	.0008	.0000
12	.0000	.0000	.0000	.0000	.0000	.0008	.0000
13	.0000	.0000	.0000	.0000	.0000	.0000	.0000
14	.0000	.0000	.0000	.0000	.0000	.0000	.0000
15	.0000	.0000	.0000	.0000	.0000	.0008	.0000

Table IV - DISTRIBUTION OF JOB STEPS PER CLASS

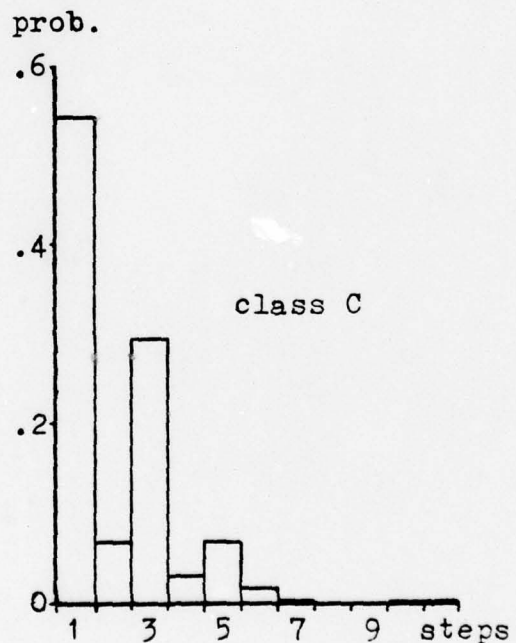
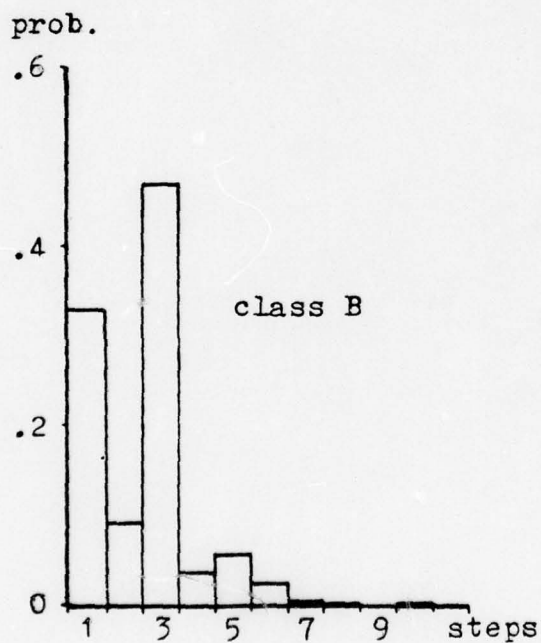
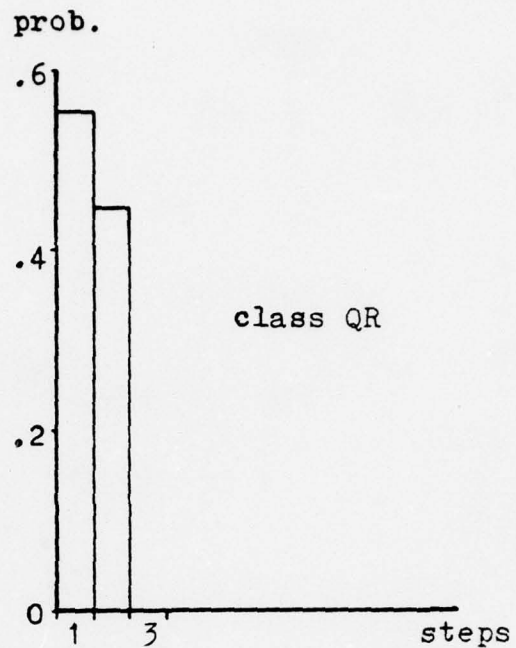
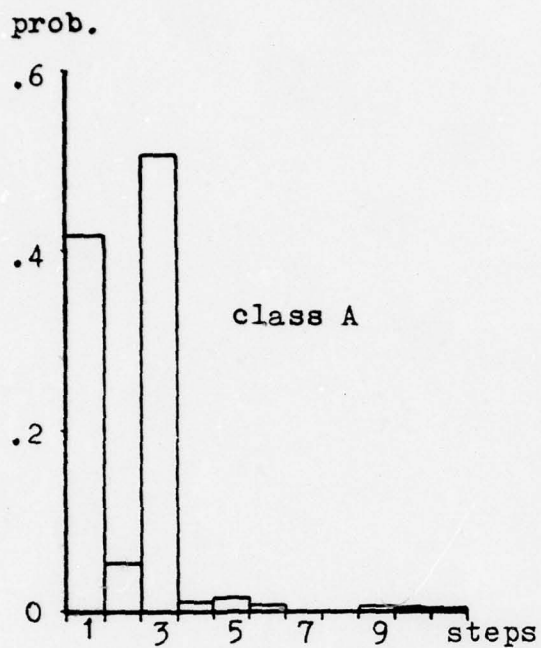


Figure 12 - HISTOGRAM: JOB STEPS PER CLASS (1)

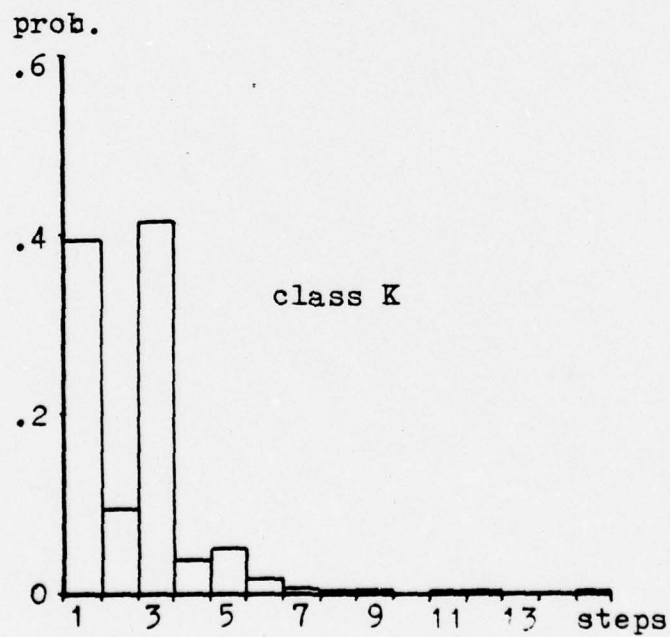
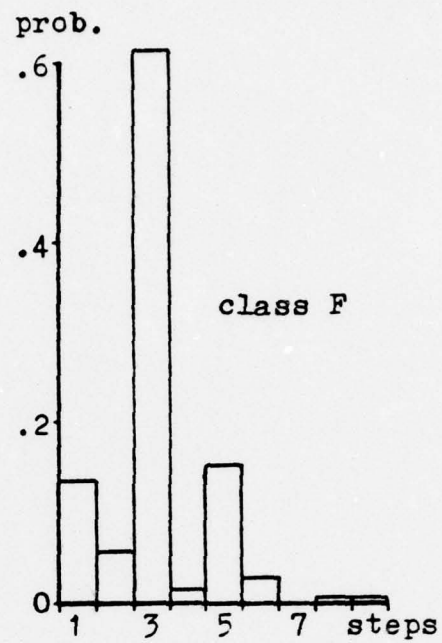
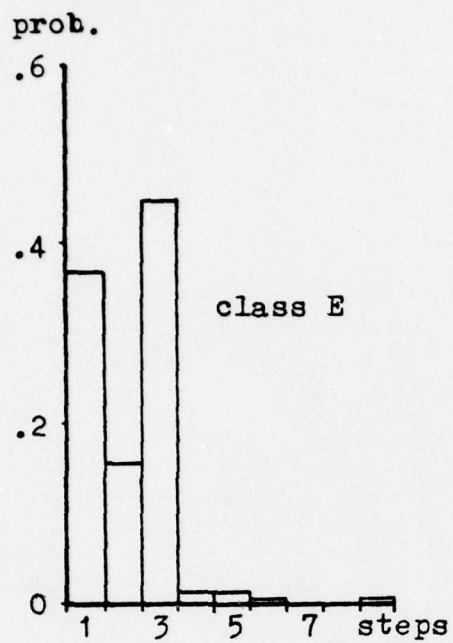


Figure 13 - HISTOGRAM: JOB STEPS PER CLASS (2)

CLASS A, QR			CLASS B			CLASS C		
CARDS		PROB.	CARDS		PROB.	CARDS		PROB.
0 -	39	.3385	0 -	39	.4004	0 -	39	.4205
40 -	79	.1686	40 -	79	.1271	40 -	79	.1646
80 -	119	.0807	80 -	119	.0727	80 -	119	.0752
120 -	159	.0687	120 -	159	.0467	120 -	159	.0659
160 -	199	.0593	160 -	199	.0486	160 -	199	.0410
200 -	239	.0617	200 -	239	.0409	200 -	239	.0232
240 -	279	.0415	240 -	279	.0417	240 -	279	.0162
280 -	319	.0307	280 -	319	.0248	280 -	319	.0381
320 -	359	.0247	320 -	359	.0180	320 -	359	.0109
360 -	399	.0244	360 -	399	.0307	360 -	399	.0113
400 -	439	.0202	400 -	439	.0135	400 -	439	.0116
440 -	479	.0107	440 -	479	.0103	440 -	479	.0251
480 -	519	.0103	480 -	519	.0128	480 -	519	.0077
520 -	559	.0063	520 -	559	.0124	520 -	559	.0086
560 -	599	.0055	560 -	599	.0086	560 -	599	.0073
600 -	639	.0029	600 -	639	.0088	600 -	639	.0069
640 -	679	.0040	640 -	679	.0055	640 -	679	.0083
680 -	719	.0043	680 -	719	.0086	680 -	719	.0036
720 -	759	.0025	720 -	759	.0027	720 -	759	.0017
760 -	799	.0051	760 -	799	.0071	760 -	799	.0023
800 -	839	.0058	800 -	839	.0057	800 -	839	.0033
840 -	879	.0023	840 -	879	.0032	840 -	879	.0043
880 -	919	.0040	880 -	919	.0035	880 -	919	.0056
920 -	959	.0043	920 -	959	.0051	920 -	959	.0030
960 -	999	.0036	960 -	999	.0040	960 -	999	.0033
1000 -	1039	.0027	1000 -	1039	.0046	1000 -	1039	.0077
1040 -	1079	.0027	1040 -	1079	.0059	1040 -	1079	.0129
1080 -	1119	.0018	1080 -	1119	.0065	1080 -	1119	.0073
1120 -	1159	.0017	1120 -	1159	.0042	1120 -	1159	.0000
1160 -	1199	.0005	1160 -	1199	.0084	1160 -	1199	.0026

Table V - DISTRIBUTION OF INPUT CARDS PER JOB (1)

CLASS E			CLASS F			CLASS K		
CARDS		PROB.	CARDS		PROB.	CARDS		PROB.
80 - 159		.2616	80 - 159		.1558	80 - 159		.0930
160 - 239		.0846	160 - 239		.0725	160 - 239		.0854
240 - 319		.0307	240 - 319		.0616	240 - 319		.0431
320 - 399		.0577	320 - 399		.0362	320 - 399		.0423
400 - 479		.0847	400 - 479		.0580	400 - 479		.0377
480 - 559		.0423	480 - 559		.0217	480 - 559		.0169
560 - 639		.0461	560 - 639		.0217	560 - 639		.0100
640 - 719		.0231	640 - 719		.0290	640 - 719		.0069
720 - 799		.0115	720 - 799		.0399	720 - 799		.0170
800 - 879		.0000	800 - 879		.1159	800 - 879		.0184
880 - 959		.0193	880 - 959		.0218	880 - 959		.0108
960 - 1039		.0000	960 - 1039		.0181	960 - 1039		.0085
1040 - 1119		.0115	1040 - 1119		.0290	1040 - 1119		.0084
1120 - 1199		.0038	1120 - 1199		.0289	1120 - 1199		.0100
1200 - 1279		.0077	1200 - 1279		.0327	1200 - 1279		.0110
1280 - 1359		.0000	1280 - 1359		.0072	1280 - 1359		.0021
1360 - 1439		.0000	1360 - 1439		.0399	1360 - 1439		.0038
1440 - 1519		.0039	1440 - 1519		.0000	1440 - 1519		.0039
1520 - 1599		.0077	1520 - 1599		.0217	1520 - 1599		.0077
1600 - 1679		.0115	1600 - 1679		.0326	1600 - 1679		.0069
1680 - 1759		.0077	1680 - 1759		.0000	1680 - 1759		.0031
1760 - 1839		.0000	1760 - 1839		.0000	1760 - 1839		.0046
1840 - 1919		.0039	1840 - 1919		.0000	1840 - 1919		.0046
1920 - 1999		.0000	1920 - 1999		.0000	1920 - 1999		.0023
2000 - 2079		.0000	2000 - 2079		.0000	2000 - 2079		.0008
2080 - 2159		.0000	2080 - 2159		.0000	2080 - 2159		.0015
2160 - 2239		.0000	2160 - 2239		.0000	2160 - 2239		.0008
2240 - 2319		.0038	2240 - 2319		.0000	2240 - 2319		.0023
2320 - 2399		.0000	2320 - 2399		.0000	2320 - 2399		.0000

Table VI - DISTRIBUTION OF INPUT CARDS PER JOB (2)

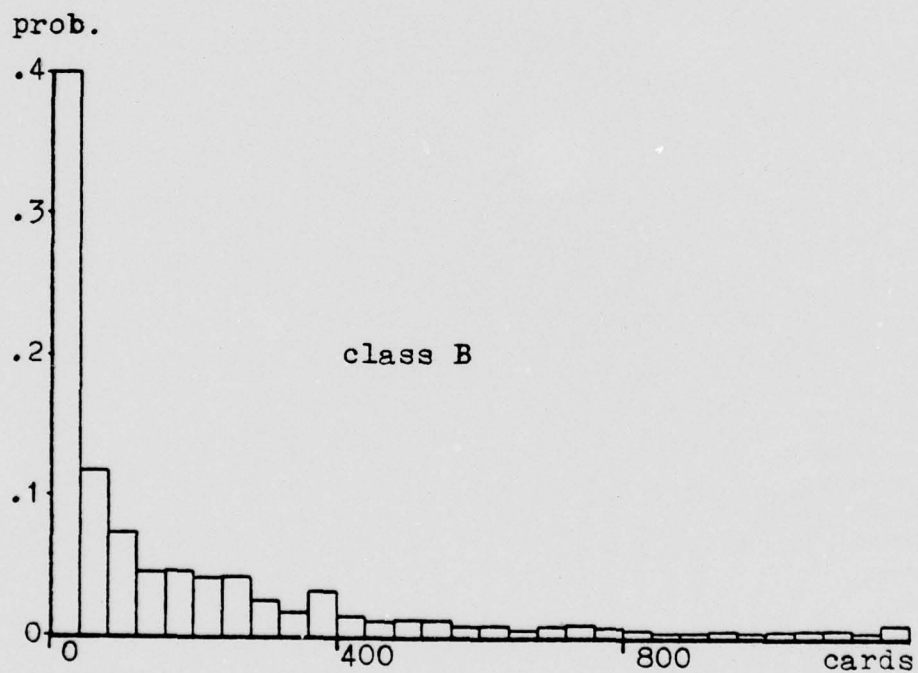
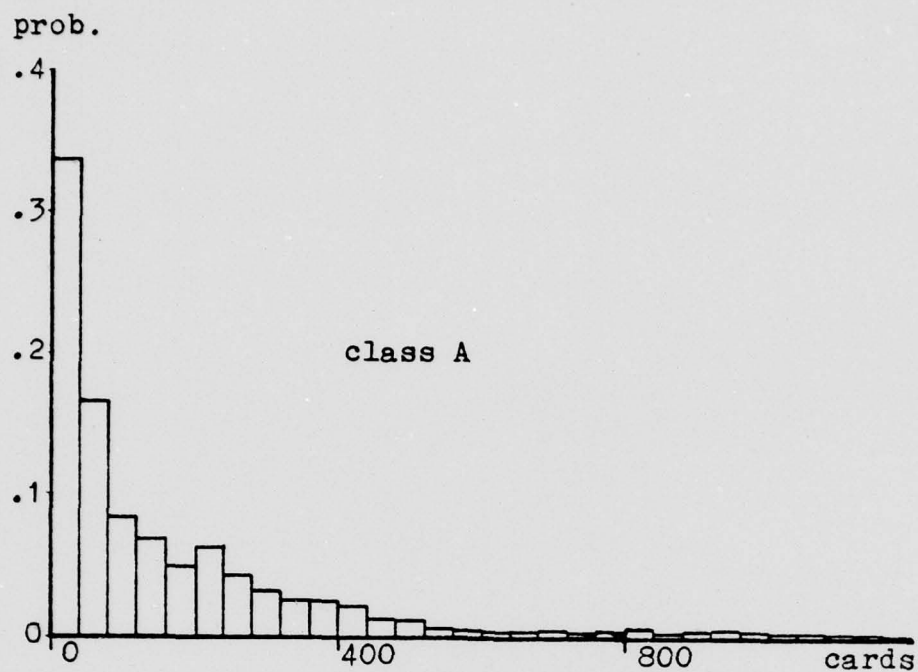


Figure 14 - HISTOGRAM: INPUT CARDS PER JOB (1)

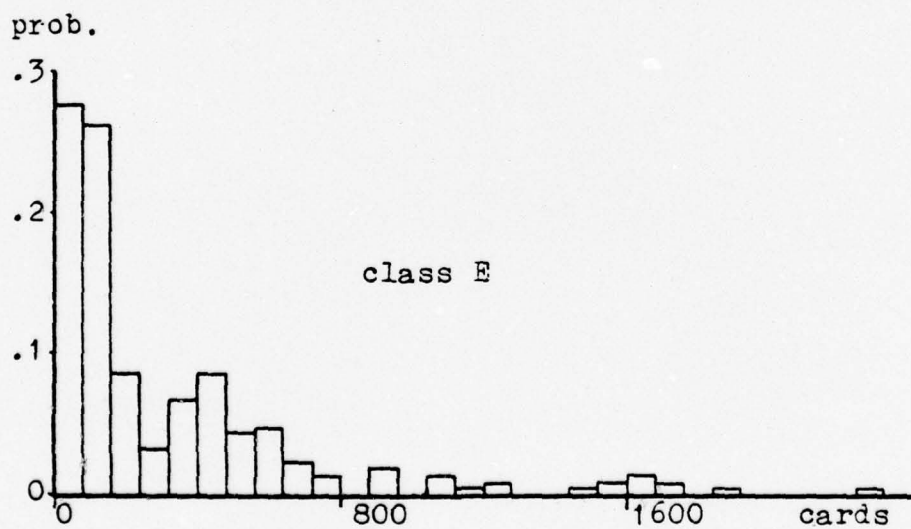
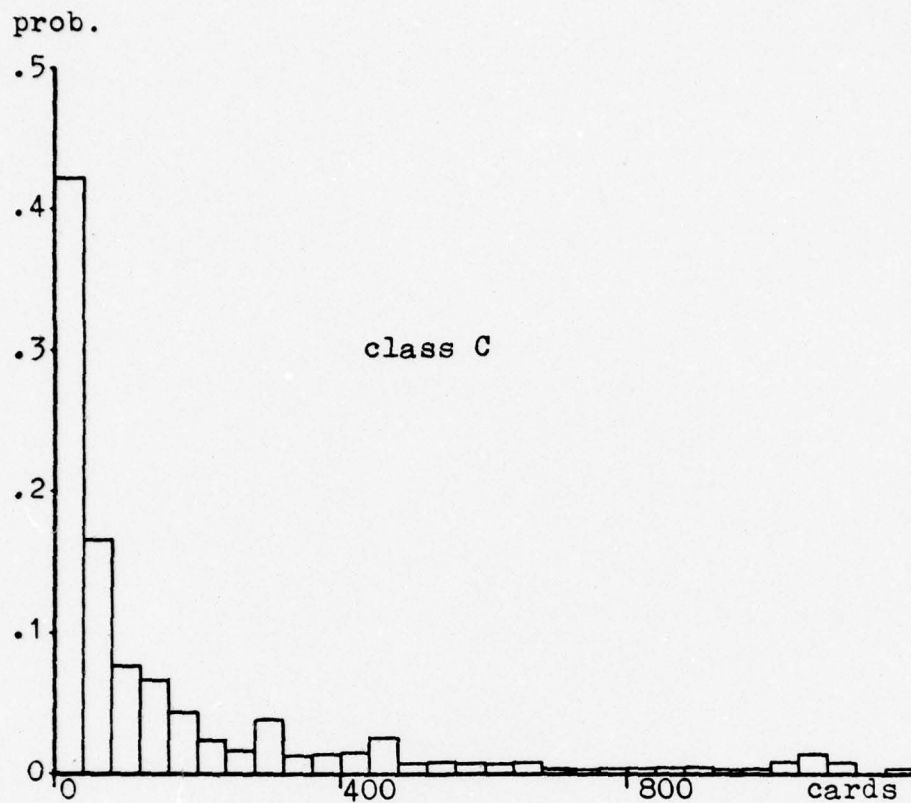


Figure 15 - HISTOGRAM: INPUT CARDS PER JOB (2)

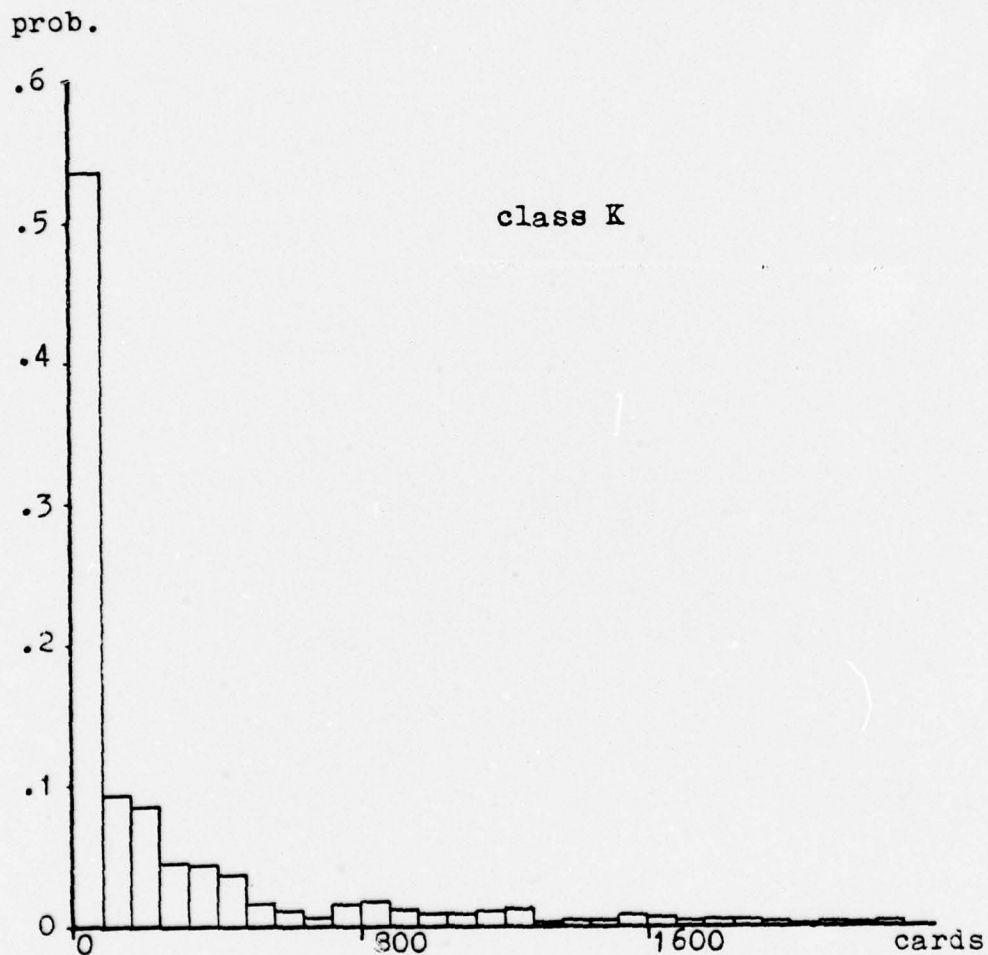
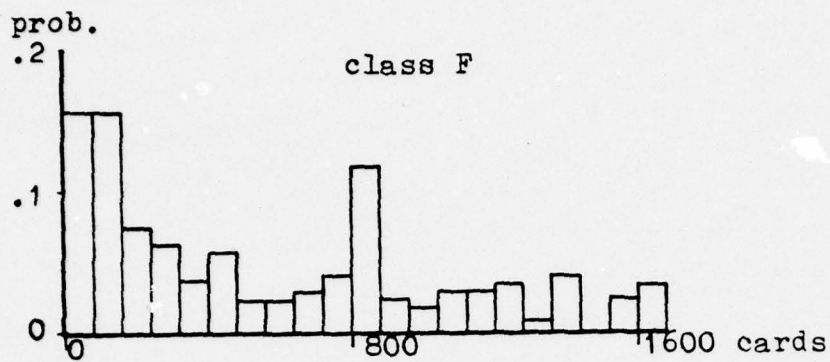


Figure 16 - HISTOGRAM: INPUT CARDS PER JOB (3)

CLASS A,QR			CLASS B			CLASS C		
CORE		PROB.	CORE		PROB.	CORE		PROB.
0 -	9	.0000	0 -	9	.0000	0 -	19	.0000
10 -	19	.0000	10 -	19	.0000	20 -	39	.0000
20 -	29	.0000	20 -	29	.0000	40 -	59	.0000
30 -	39	.0000	30 -	39	.0000	60 -	79	.1516
40 -	49	.0000	40 -	49	.0000	80 -	99	.0047
50 -	59	.0000	50 -	59	.0000	100 -	119	.3282
60 -	69	.3074	60 -	69	.2049	120 -	139	.0218
70 -	79	.0044	70 -	79	.0007	140 -	159	.0041
80 -	89	.0031	80 -	89	.0202	160 -	179	.1330
90 -	99	.0079	90 -	99	.0069	180 -	199	.0333
100 -	109	.5039	100 -	109	.4867	200 -	219	.1227
110 -	119	.0055	110 -	119	.0209	220 -	239	.0156
120 -	129	.0099	120 -	129	.0236	240 -	259	.1850
130 -	139	.0062	130 -	139	.0079	260 -	279	.0000
140 -	149	.0002	140 -	149	.0007	280 -	299	.0000
150 -	159	.1133	150 -	159	.1428	300 -	319	.0000
160 -	169	.0041	160 -	169	.0086	320 -	339	.0000
170 -	179	.0045	170 -	179	.0138	340 -	359	.0000
180 -	189	.0296	180 -	189	.0623	360 -	379	.0000

min. core size (all classes): 62 K

max. core size (class A,QR): 180 K

max. core size (class B): 180 K

max. core size (class C): 250 K

Table VII - DISTRIBUTION OF CORE USED PER STEP (1)

CLASS E			CLASS F			CLASS K		
CORE		PROB.	CORE		PROB.	CORE		PROB.
0 - 39		.0000	0 - 39		.0000	0 - 39		.0000
40 - 79		.1238	40 - 79		.1688	40 - 79		.1851
80 - 119		.3254	80 - 119		.4087	80 - 119		.3978
120 - 159		.0032	120 - 159		.0321	120 - 159		.0501
160 - 199		.0333	160 - 199		.2032	160 - 199		.1599
200 - 239		.0317	200 - 239		.0149	200 - 239		.0465
240 - 279		.0556	240 - 279		.0700	240 - 279		.0633
280 - 319		.2302	280 - 319		.0539	280 - 319		.0255
320 - 359		.1968	320 - 359		.0000	320 - 359		.0029
360 - 399		.0000	360 - 399		.0229	360 - 399		.0155
400 - 439		.0000	400 - 439		.0255	400 - 439		.0149
440 - 479		.0000	440 - 479		.0000	440 - 479		.0081
480 - 519		.0000	480 - 519		.0000	480 - 519		.0032
520 - 559		.0000	520 - 559		.0000	520 - 559		.0000
560 - 599		.0000	560 - 599		.0000	560 - 599		.0175
600 - 639		.0000	600 - 639		.0000	600 - 639		.0061
640 - 679		.0000	640 - 679		.0000	640 - 679		.0036
680 - 719		.0000	680 - 719		.0000	680 - 719		.0000
720 - 759		.0000	720 - 759		.0000	720 - 759		.0000

min. core size (all classes): 62 K
 max. core size (class E): 350 K
 max. core size (class F): 400 K
 max. core size (class K): >400 K

Table VIII - DISTRIBUTION OF CORE USED PER STEP (2)

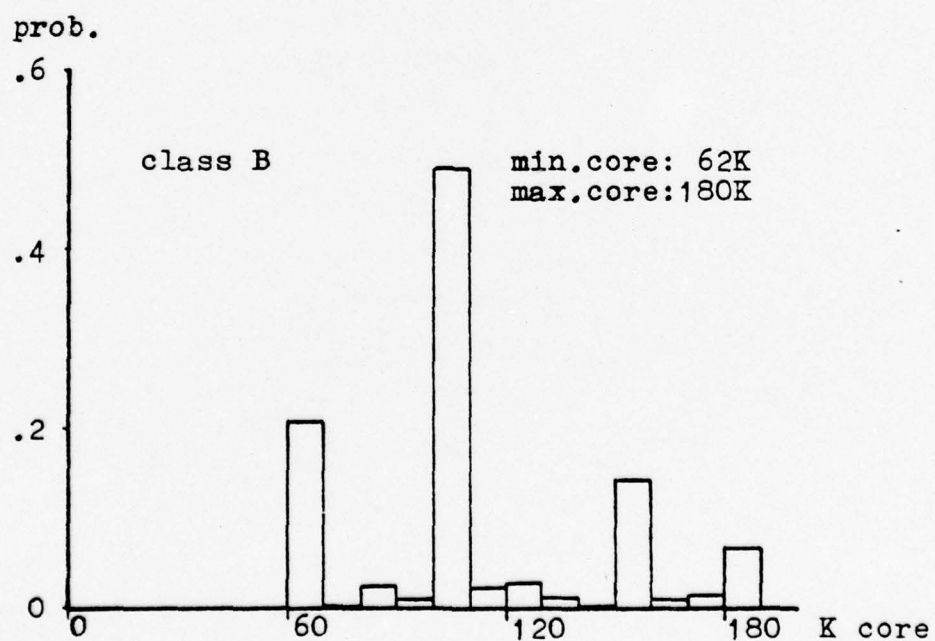
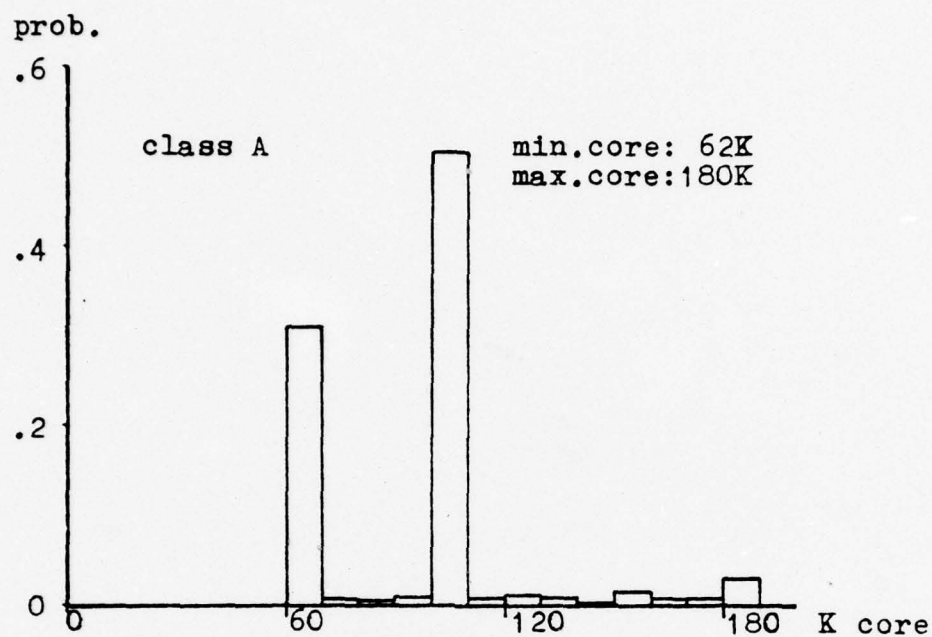


Figure 17 - HISTOGRAM: CORE USED PER STEP (1)

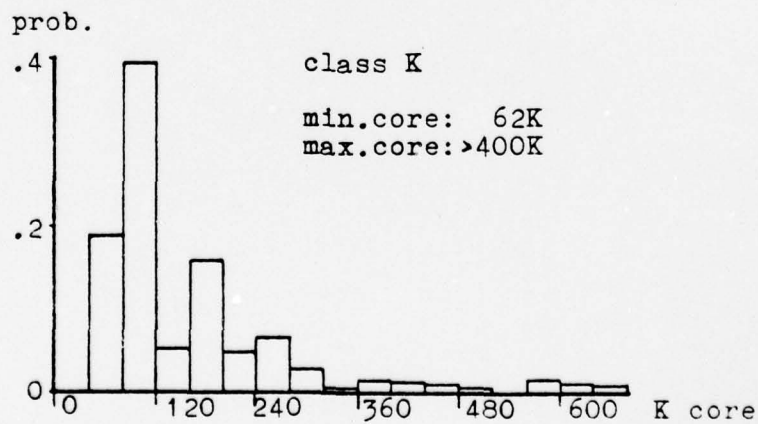
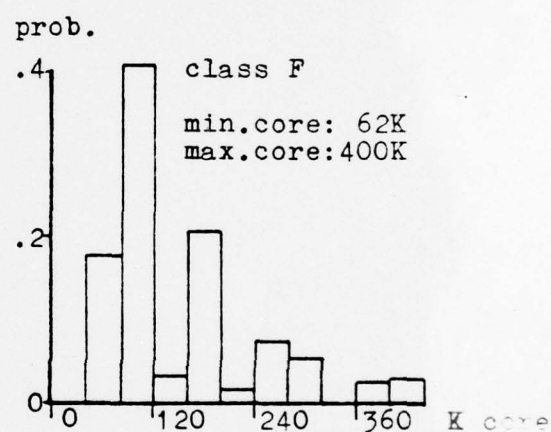
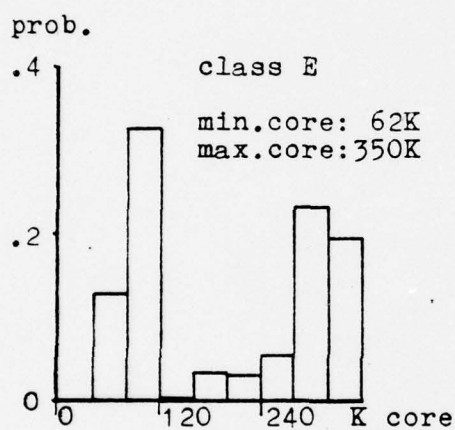
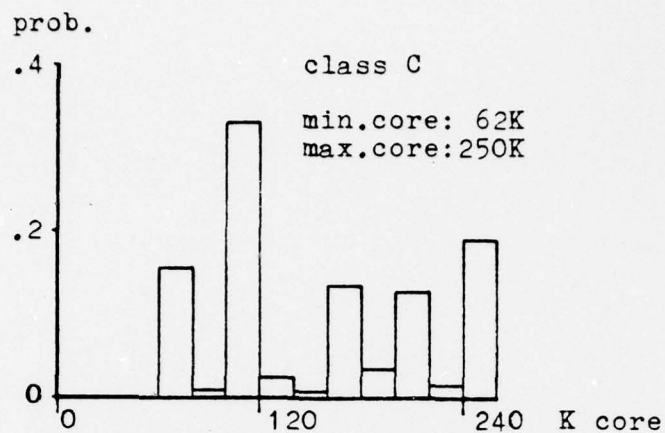


Figure 18 - HISTOGRAM: CORE USED PER STEP (2)

CLASS A, QR			CLASS B		CLASS C		
SECONDS		PROB.	SECONDS	PROB.	SECONDS		PROB.
0 - 9		.0621	0 - 19	.0962	0 - 39		.1922
10 - 19		.0630	20 - 39	.1304	40 - 79		.2134
20 - 29		.0979	40 - 59	.1359	80 - 119		.1359
30 - 39		.1128	60 - 79	.1210	120 - 159		.0925
40 - 49		.1048	90 - 99	.0963	160 - 199		.0579
50 - 59		.0943	100 - 119	.0774	200 - 239		.0454
60 - 69		.0790	120 - 139	.0549	240 - 279		.0357
70 - 79		.0595	140 - 159	.0445	280 - 319		.0289
80 - 89		.0511	160 - 179	.0371	320 - 359		.0266
90 - 99		.0437	180 - 199	.0298	360 - 399		.0194
100 - 109		.0367	200 - 219	.0243	400 - 439		.0202
110 - 119		.0307	220 - 239	.0181	440 - 479		.0145
120 - 129		.0247	240 - 259	.0164	480 - 519		.0150
130 - 139		.0210	260 - 279	.0143	520 - 559		.0122
140 - 149		.0170	280 - 299	.0124	560 - 599		.0098
150 - 159		.0132	300 - 319	.0092	600 - 639		.0078
160 - 169		.0123	320 - 339	.0090	640 - 679		.0069
170 - 179		.0096	340 - 359	.0068	680 - 719		.0067
180 - 189		.0089	360 - 379	.0058	720 - 759		.0061
190 - 199		.0064	380 - 399	.0061	760 - 799		.0057
200 - 209		.0063	400 - 419	.0053	800 - 839		.0050
210 - 219		.0051	420 - 439	.0047	840 - 879		.0047
220 - 229		.0047	440 - 459	.0050	880 - 919		.0051
230 - 239		.0036	460 - 479	.0040	920 - 959		.0058
240 - 249		.0027	480 - 499	.0032	960 - 999		.0037
250 - 259		.0032	500 - 519	.0034	1000 - 1039		.0029
260 - 269		.0028	520 - 539	.0036	1040 - 1079		.0026
270 - 279		.0023	540 - 559	.0031	1080 - 1119		.0023
280 - 289		.0021	560 - 579	.0025	1120 - 1159		.0021
290 - 299		.0018	580 - 599	.0028	1160 - 1199		.0009
300 - 309		.0019	600 - 619	.0018	1200 - 1239		.0014
310 - 319		.0016	620 - 639	.0021	1240 - 1279		.0012
320 - 329		.0016	640 - 659	.0018	1280 - 1319		.0017
330 - 339		.0015	660 - 679	.0021	1320 - 1359		.0021
340 - 349		.0013	680 - 699	.0014	1360 - 1399		.0016
350 - 359		.0017	700 - 719	.0018	1400 - 1439		.0009
360 - 369		.0012	720 - 739	.0014	1440 - 1479		.0017
370 - 379		.0009	740 - 759	.0014	1480 - 1519		.0015
380 - 389		.0013	760 - 779	.0013	1520 - 1559		.0000
390 - 399		.0012	780 - 799	.0014	1560 - 1599		.0000
400 - 409		.0005	800 - 819	.0000	1600 - 1639		.0000
410 - 419		.0010	820 - 839	.0000	1640 - 1679		.0000
420 - 429		.0006	840 - 859	.0000	1680 - 1719		.0000
430 - 439		.0004	860 - 879	.0000	1720 - 1759		.0000
440 - 449		.0000	880 - 899	.0000	1760 - 1799		.0000

Table IX - DISTRIBUTION OF ELAPSED STEP RUN TIME (1)

CLASS E			CLASS F			CLASS K		
SECONDS		PROB.	SECONDS		PROB.	SECONDS		PROB.
0 - 39		.2107	0 - 39		.3698	0 - 79		.4677
40 - 79		.2593	40 - 79		.2117	80 - 159		.1680
80 - 119		.1313	80 - 119		.0998	160 - 239		.0710
120 - 159		.0778	120 - 159		.0425	240 - 319		.0512
160 - 199		.0600	160 - 199		.0402	320 - 399		.0357
200 - 239		.0421	200 - 239		.0280	400 - 479		.0229
240 - 279		.0276	240 - 279		.0170	480 - 559		.0151
280 - 319		.0275	280 - 319		.0195	560 - 639		.0341
320 - 359		.0173	320 - 359		.0097	640 - 719		.0151
360 - 399		.0195	360 - 399		.0158	720 - 799		.0138
400 - 439		.0129	400 - 439		.0122	800 - 879		.0121
440 - 479		.0130	440 - 479		.0146	880 - 959		.0078
480 - 519		.0130	480 - 519		.0146	960 - 1039		.0087
520 - 559		.0129	520 - 559		.0146	1040 - 1119		.0091
560 - 599		.0081	560 - 599		.0085	1120 - 1199		.0084
600 - 639		.0065	600 - 639		.0122	1200 - 1279		.0058
640 - 679		.0081	640 - 679		.0121	1280 - 1359		.0064
680 - 719		.0049	680 - 719		.0110	1360 - 1439		.0053
720 - 759		.0081	720 - 759		.0061	1440 - 1519		.0041
760 - 799		.0032	760 - 799		.0036	1520 - 1599		.0054
800 - 839		.0081	800 - 839		.0061	1600 - 1679		.0033
840 - 879		.0033	840 - 879		.0036	1680 - 1759		.0031
880 - 919		.0064	880 - 919		.0061	1760 - 1839		.0040
920 - 959		.0049	920 - 959		.0049	1840 - 1919		.0047
960 - 999		.0068	960 - 999		.0049	1920 - 1999		.0031
1000 - 1039		.0149	1000 - 1039		.0012	2000 - 2079		.0023
1040 - 1079		.0032	1040 - 1079		.0000	2080 - 2159		.0027
1080 - 1119		.0017	1080 - 1119		.0036	2160 - 2239		.0037
1120 - 1159		.0000	1120 - 1159		.0037	2240 - 2319		.0020
1160 - 1199		.0000	1160 - 1199		.0024	2320 - 2399		.0034

Table X - DISTRIBUTION OF ELAPSED STEP RUN TIME (2)

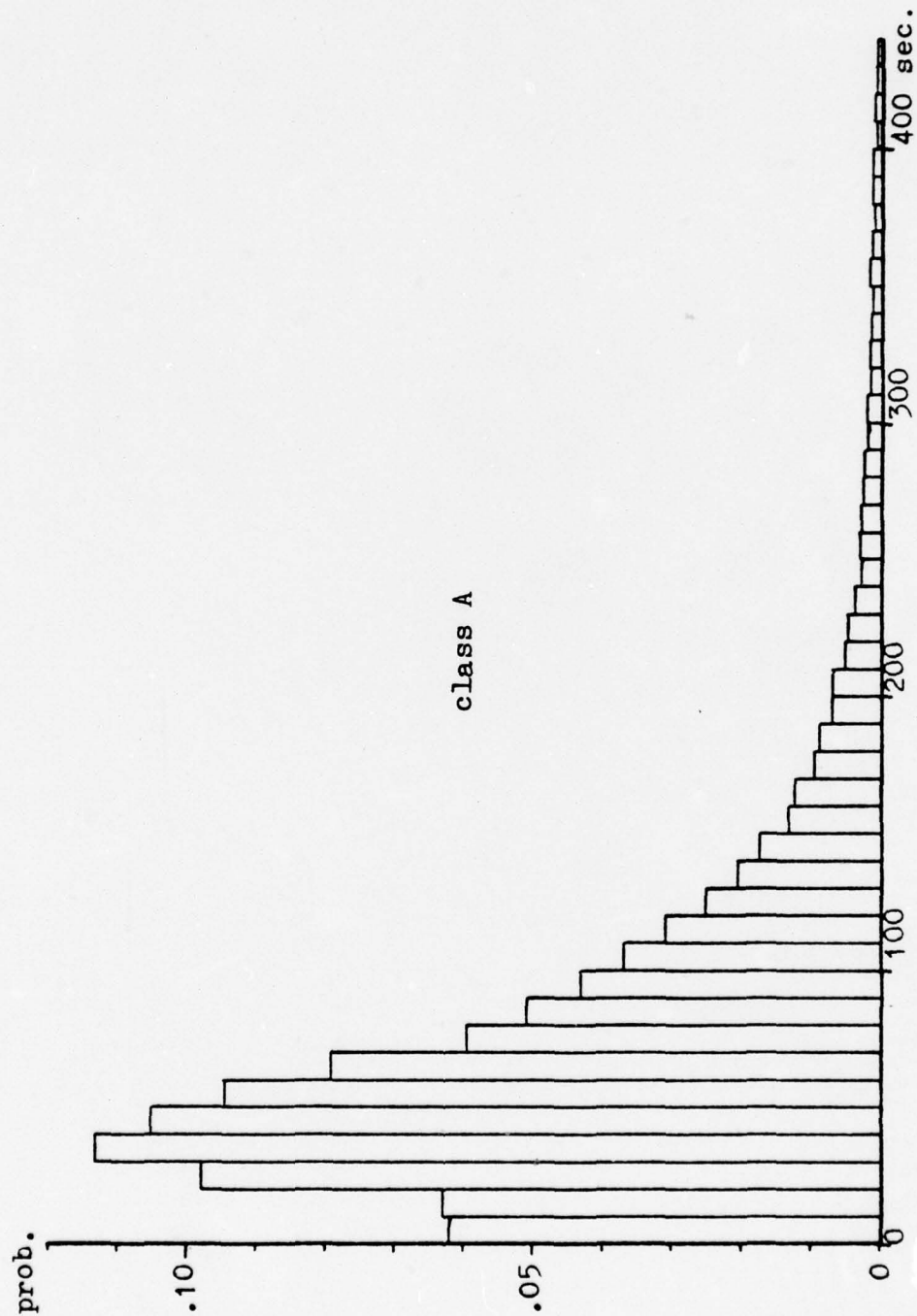


Figure 19 - HISTOGRAM: ELAPSED STEP RUN TIME (1)

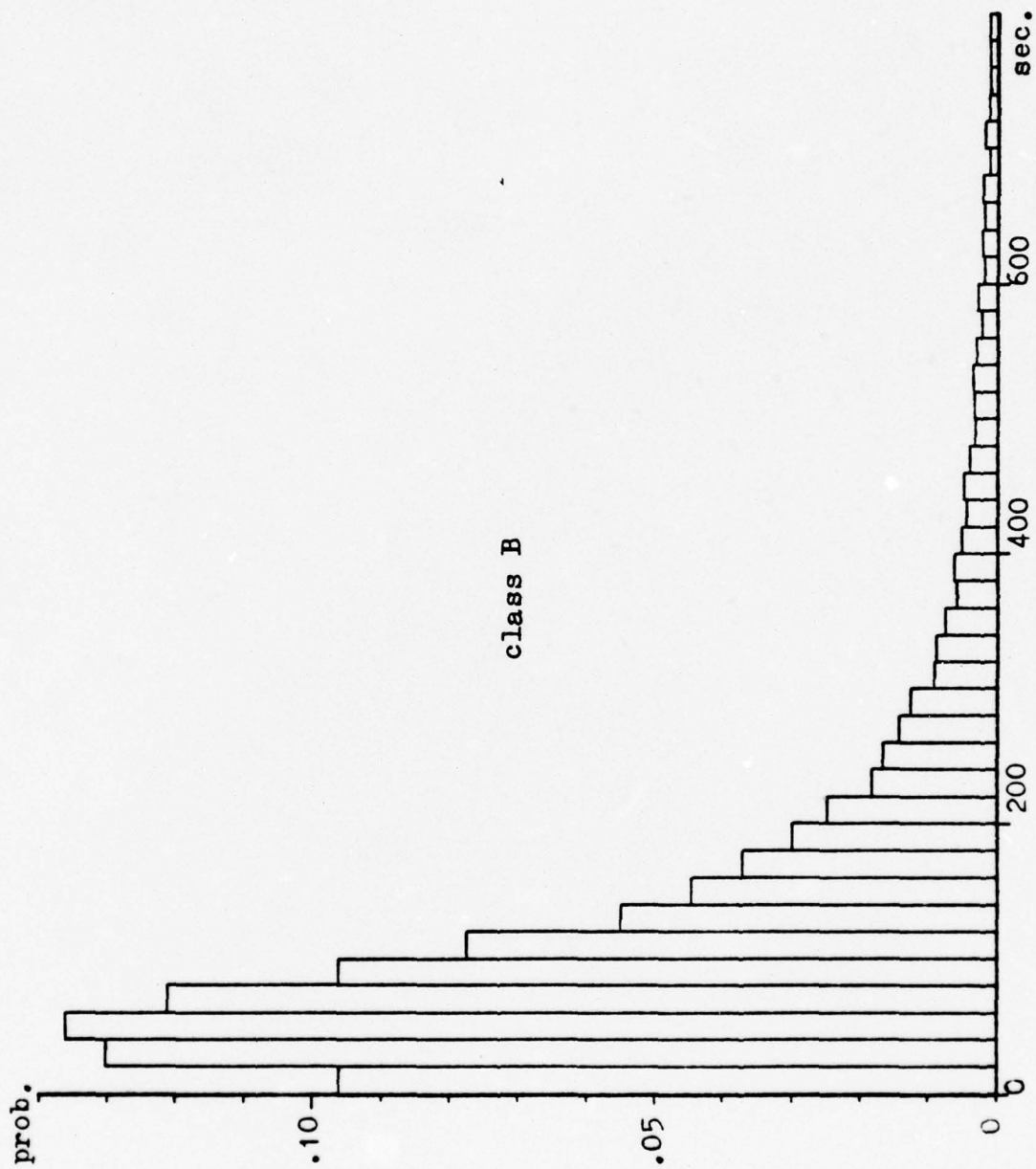


Figure 20 - HISTOGRAM: ELAPSED STEP RUN TIME (2)

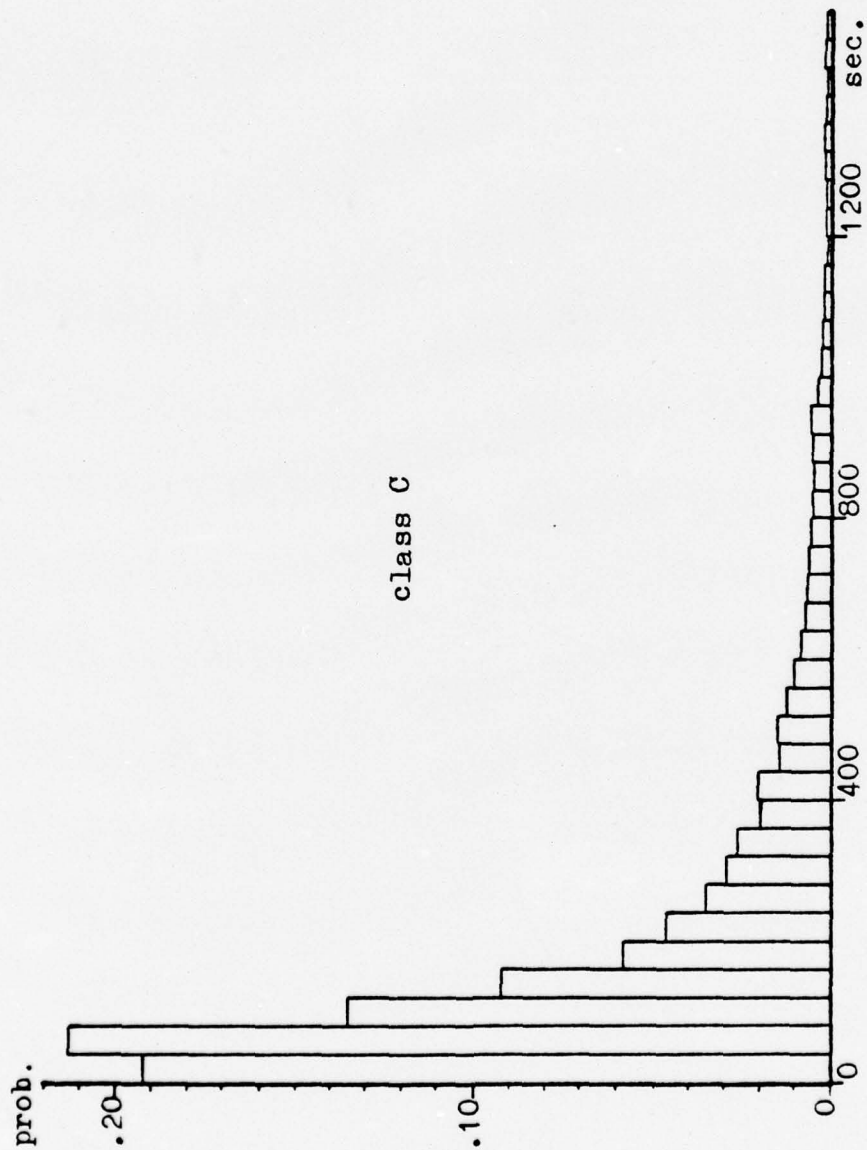


Figure 21 - HISTOGRAM: ELAPSED STEP RUN TIME (3)

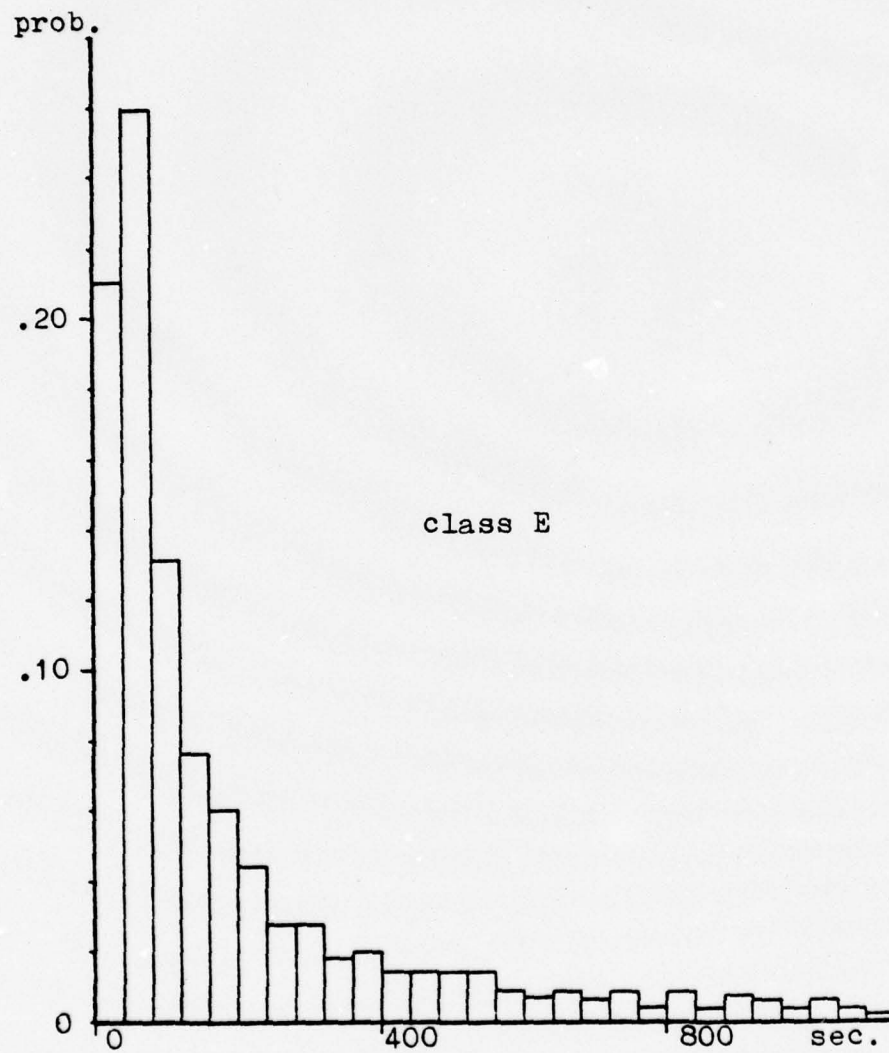


Figure 22 - HISTOGRAM: ELAPSED STEP RUN TIME (4)

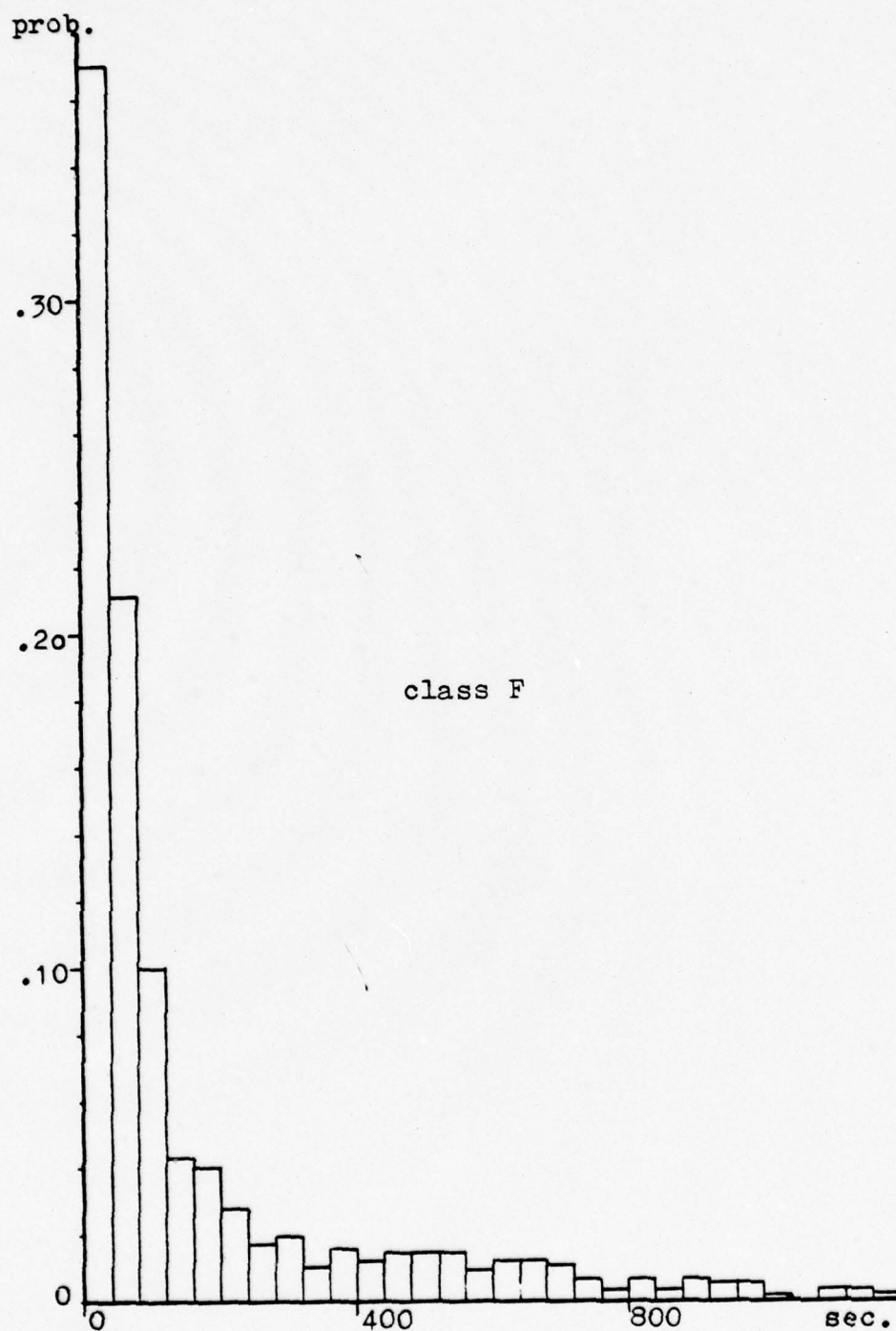


Figure 23 - HISTOGRAM: ELAPSED STEP RUN TIME (5)

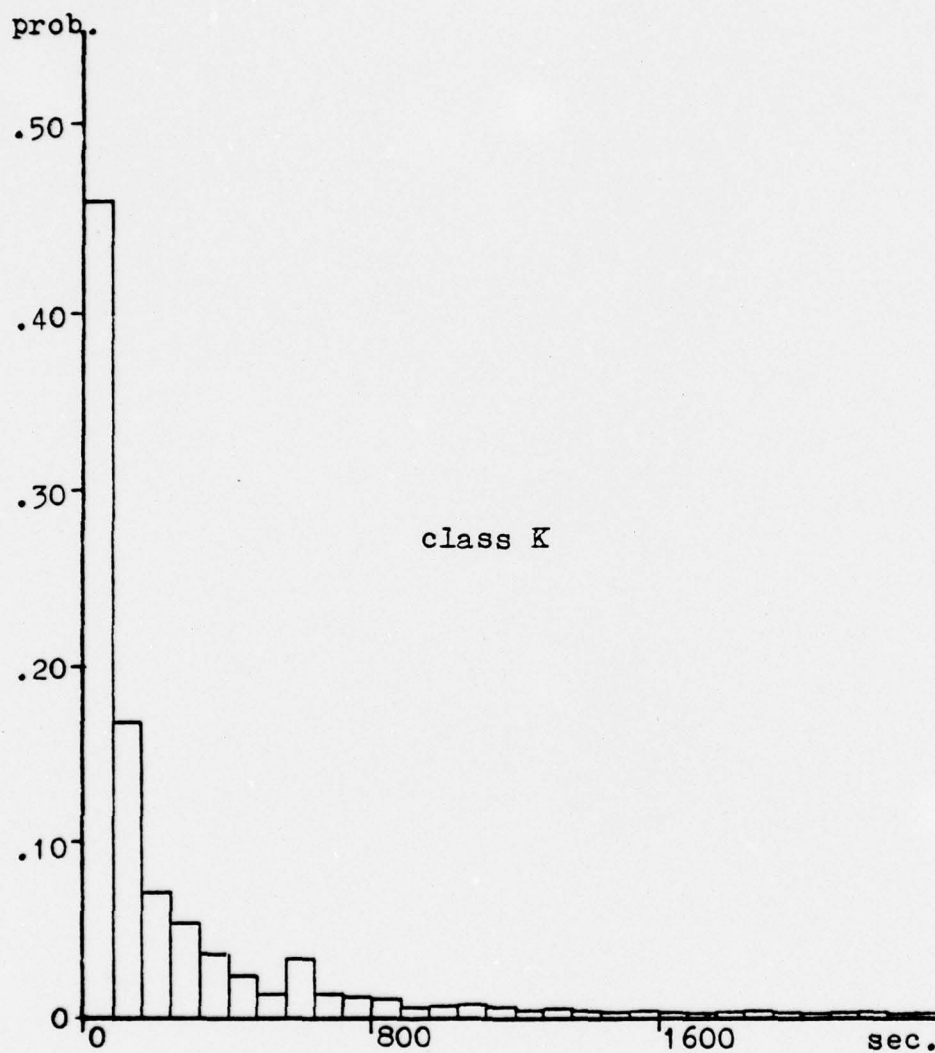


Figure 24 - HISTOGRAM: ELAPSED STEP RUN TIME (6)

		CLASS			
TAPES	DISKS	B	C	E	K
0	0	.8887	.8887	.8887	.8570
0	1	.0395	.0395	.0395	.0381
0	2	.0019	.0019	.0019	.0018
1	1	.0019	.0019	.0019	.0018
2	1	.0005	.0005	.0005	.0005
2	2	.0003	.0003	.0003	.0003
1	3	.0003	.0003	.0003	.0003
2	3	.0005	.0005	.0005	.0005
1	0	.0584	.0584	.0584	.0563
2	0	.0079	.0079	.0079	.0076
3	0	.0000	.0000	.0000	.0357

Table XI - DISTRIBUTION OF TAPES AND DISKS PER JOB CLASS

C. OPERATOR RESPONSE TIMES

The system logs were also used to evaluate the operator volume mounting times, their response times to other system requests, and the number of jobs cancelled by the operators because a request could not be satisfied. One problem for the evaluation was the fact that the system requests had no time stamps. In most cases the time could be estimated within a 10 second range from other system messages with time stamps just above and below the request messages. For tape and disk mounts there were also no direct operator answers on the system logs, but in a certain number of cases the actual mounting time could be estimated from other system messages. Here again only those cases were evaluated where the estimation could be made within a 10 second time range. Using this approach a total of about 700 operator response times could be used. The probability distribution per job step, separated into the cases for tape mount, disk mount, and other system requests, is given in Table XII; a histogram is provided in Figure 25.

The relatively high probability of short reaction times to tape and disk mount requests came from the fact that the requested volumes were already pre-mounted and the devices had only to be varied on-line.

The number of jobs cancelled by the operators because a request could not be satisfied could be counted exactly: 49 jobs or 1.31 % out of 3,735 jobs.

TAPE MOUNT REQUESTS			DISK MOUNT REQUESTS			OTHER REQUESTS		
SECONDS		PROB.	SECONDS		PROB.	SECONDS		PROB.
0	- 19	.0000	0	- 19	.0000	0	- 19	.6715
20	- 39	.0733	20	- 39	.0197	20	- 39	.0253
40	- 59	.1721	40	- 59	.0527	40	- 59	.0433
60	- 79	.1795	60	- 79	.0789	60	- 79	.0577
80	- 99	.1172	80	- 99	.1316	80	- 99	.0181
100	- 119	.0843	100	- 119	.0658	100	- 119	.0108
120	- 139	.0732	120	- 139	.1381	120	- 139	.0217
140	- 159	.0257	140	- 159	.0856	140	- 159	.0142
160	- 179	.0256	160	- 179	.0526	160	- 179	.0074
180	- 199	.0330	180	- 199	.0592	180	- 199	.0217
200	- 219	.0329	200	- 219	.0592	200	- 219	.0036
220	- 239	.0184	220	- 239	.0132	220	- 239	.0037
240	- 259	.0329	240	- 259	.0395	240	- 259	.0036
260	- 279	.0147	260	- 279	.0131	260	- 279	.0036
280	- 299	.0110	280	- 299	.0263	280	- 299	.0036
300	- 319	.0183	300	- 319	.0132	300	- 319	.0036
320	- 339	.0146	320	- 339	.0131	320	- 339	.0036
340	- 359	.0110	340	- 359	.0198	340	- 359	.0036
360	- 379	.0074	360	- 379	.0263	360	- 379	.0036
380	- 399	.0073	380	- 399	.0197	380	- 399	.0036
400	- 419	.0027	400	- 419	.0066	400	- 419	.0036
420	- 439	.0173	420	- 439	.0061	420	- 439	.0036
440	- 459	.0037	440	- 459	.0071	440	- 459	.0037
460	- 479	.0037	460	- 479	.0065	460	- 479	.0036
480	- 499	.0036	480	- 499	.0066	480	- 499	.0036
500	- 519	.0037	500	- 519	.0066	500	- 519	.0036
520	- 539	.0036	520	- 539	.0066	520	- 539	.0036
540	- 559	.0037	540	- 559	.0066	540	- 559	.0036
560	- 579	.0037	560	- 579	.0065	560	- 579	.0036
580	- 599	.0036	580	- 599	.0066	580	- 599	.0036
600	- 619	.0037	600	- 619	.0066	600	- 619	.0036
620	- 639	.0000	620	- 639	.0000	620	- 639	.0036
640	- 659	.0000	640	- 659	.0000	640	- 659	.0036
660	- 679	.0000	660	- 679	.0000	660	- 679	.0037
680	- 699	.0000	680	- 699	.0000	680	- 699	.0036
700	- 719	.0000	700	- 719	.0000	700	- 719	.0036
720	- 739	.0000	720	- 739	.0000	720	- 739	.0036
740	- 759	.0000	740	- 759	.0000	740	- 759	.0036
760	- 779	.0000	760	- 779	.0000	760	- 779	.0036
780	- 799	.0000	780	- 799	.0000	780	- 799	.0036

Table XII - DISTRIBUTION OF OPERATOR RESPONSE TIMES

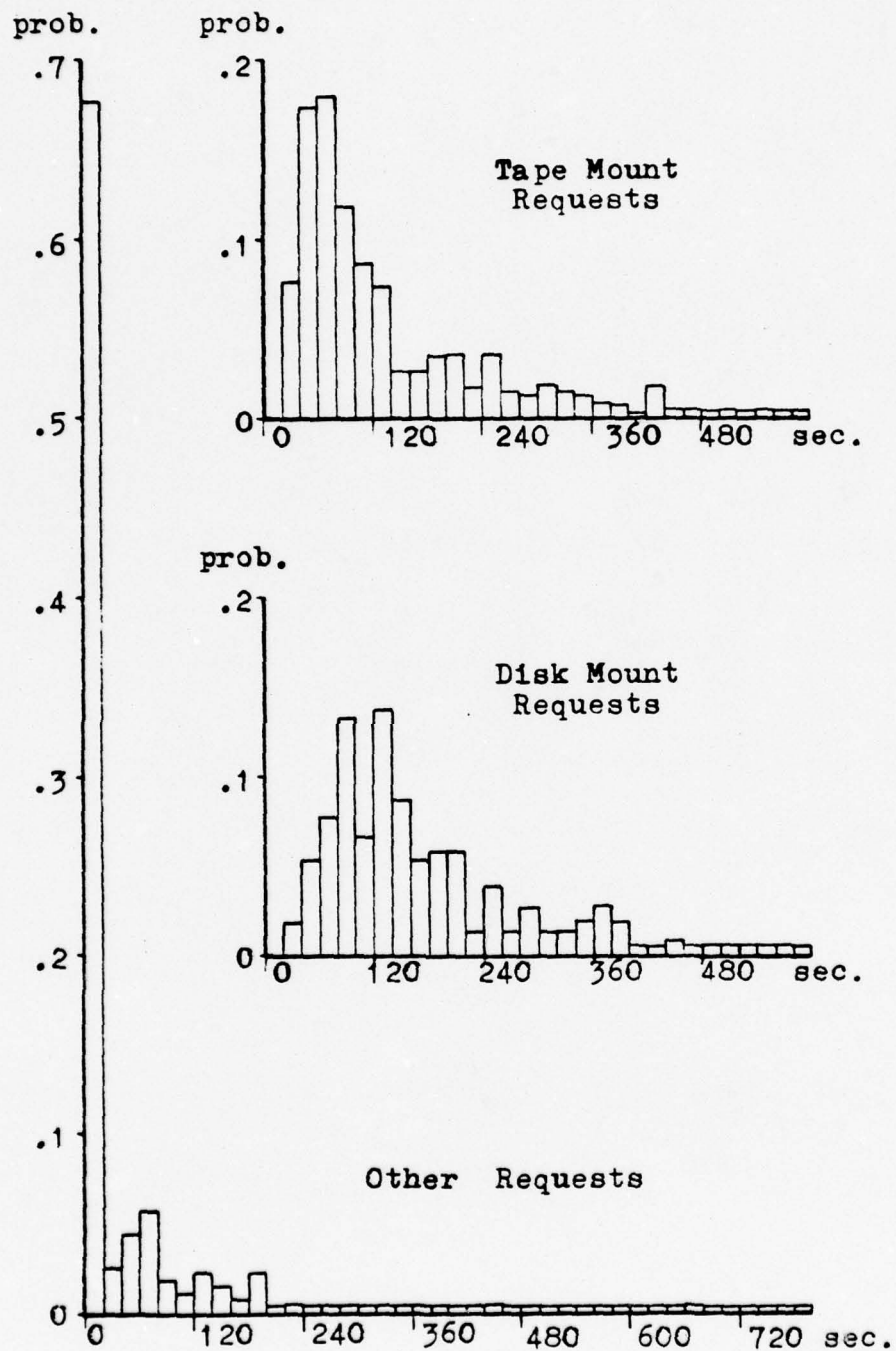


Figure 25 - HISTOGRAM: OPERATOR RESPONSE TIMES

D. SYSTEM PARAMETERS

The following system parameters were used to tailor the simulation model to the environment at the Naval Postgraduate school as it was available to the user during the time period April to August 1976:

- * number of tape drives: 9
- * number of disk drives: 3
- * amount of input spool space: 45,000 card images
- * amount of direct access space: 100 records
- * main memory (high address) : 1140 K
- * main memory (low address) : 140 K

The actual number of disk drives in the system was much higher (see Table I), but without special arrangements only three were free for general users. Also only five of the eight core boxes were routinely available for OS/MVT.

Assuming a mean of 300 input cards per jobs the amount of 45,000 card images was equivalent to the current system spooling capacity of about 150 jobs.

As mentioned earlier the direct access space was not used as parameter for the simulation runs. Thus the number of 100 records had no meaning.

The upper and lower address of main memory were the bounds of the Dynamic Area. These bounds varied depending on the load on the system. The values of the bounds used were mean values observed from the system logs.

V. VALIDATION

In order to show the usefulness and validity of the simulation model it was parameterized to match the characteristics of the computer center installation at the Naval Postgraduate School. The parameters used for input job stream characteristics, system configuration, and operator response times were mostly the same as described in the previous chapter. The outcome of the simulation runs could be compared with data observed from the actual system.

An unexpected problem arose when searching for console log data which could be compared with simulation results. Within August 1976, the only month for which both SMF tapes and system logs were available, there were 15 days which qualified for use in the model (more than 500 job arrivals in the period from 10 a.m. to 5 p.m.). At first this seemed to be a sufficient number of days to choose from, but a more detailed examination showed that none of these days could be used. For each day there was either system down time, or the operators held the queues up to 50 minutes, or both. In addition, the operators reset up to 40 jobs daily from one class into another or changed job priorities. The longest continuous time interval without down time, or queue hold, or with few resets was 4.5 hours. It was observed from 10:00 a.m. to 2:30 p.m. on August 16 1976. This was a rather short time period for validation purposes, but for lack of better data it had to be used.

The job arrival rate (1.2407 jobs per min.) and the job class distribution (see Table XIII) within this time interval differed significantly from the values observed over the three month period. The appropriate modification in the simulation model was made.

CLASS:	A	B	C	E	F	K	QR
PROB.:	.3403	.1940	.1045	.0179	.0149	.0716	.2567

Table XIII - DISTRIBUTION OF JOB CLASSES (VALIDATION RUNS)

Table XIV shows the usage of Initiators and their associated job classes during the validation runs. This set-up differed only in two minor points from the actual usage: Class O in the validation runs represented the old Quickrun class and class K was used in the validation runs instead of class M.

	TIME			
Initiator	10:00	12:00	12:18	14:30
1	oab	oab	oab	oab
2	oab	oab	oab	oab
3	oabc	oabc	oabc	oabc
4	oabc	oabc	oabc	oabc
5	oabce	oabce	oabce	oabce
6	k	kabfec	kabfec	kabfec
7	-	-	ab	ab

Table XIV - INITIATOR USAGE (VALIDATION RUNS)

Forty validation runs with different input job streams were made. A comparison between the actual values and the mean values from the simulations is given in Table XV. More jobs were started in some classes than arrived because the queues were partly filled with jobs which had arrived during the previous hour.

Actual Data:

	CLASS							
	A	B	C	E	F	K	QR	total
A:	114	65	35	6	5	24	86	335
S:	126	63	32	0	3	9	86	319
R:	1.1053	.9692	.9143	.0000	.6000	.3750	1.0000	.9522

Validation Results:

	CLASS							
	A	B	C	E	F	K	QR	total
A:	127	69	40	5	7	29	77	354
S:	131	71	39	3	1	20	7	343
R:	1.0270	1.0195	.9681	.6212	.1579	.7114	1.0083	.9689

A: Number of jobs arrived

S: Number of jobs started

R: Ratio jobs started to jobs arrived

Table XV - VALIDATION RESULTS

The ratio of jobs started to jobs arrived observed from the evaluation runs was very close to the actual ratio for the job class O (=Quickrun) and for the total. Good results were also obtained for classes A, B, and C. Since the sample size for classes E and F was small the results were meaningless. Class K results were not representative since in the actual system class M was used for K class jobs and these jobs were selected by the operator.

Due to lack of more usable data no further comparison against actual system performance could be made. The small sample size available for this kind of validation did not allow a definitive statement about the accuracy of the results.

Numerous additional validation runs have been made to check individual components of the model (region management, device allocation, etc.) and to test boundary conditions (limitation in number of devices, core size, etc.) . All of these runs showed the expected results.

However, one unusual result was observed. Although the job arrival distribution generated by the simulation model closely approximated the desired distribution for a sample size of 10,000 jobs, the job arrival rate for the first 600 jobs was always too high for a given seed. To overcome this anomaly a new feature was added to the model. Upon user's request the seed for the random number generator was modified by the value of the computer clock. It was then possible to change the seed at random. When this feature was used in additional simulation runs the unusual statistical pattern was no longer observed.

Since the future use of the simulation model is to compare the relative merits of different Initiator strategies rather than to predict absolute performance, it was sufficient to assure that the principal characteristics of the Job Management functions were reasonably well simulated. The results so far demonstrate the correct functioning of the simulation model.

APPENDIX A

USER'S MANUAL

Described in this manual is the use of the simulation model under the Control Program / Cambridge Monitor System (CP/CMS) at the Naval Postgraduate School. The user should have some private CP/CMS space (P-disk) and should be familiar with the basic functions and commands of this time-sharing system.

The examples of CP/CMS commands are from an actual run. They show how to prepare and run the model and how to get results.

TABLE OF CONTENTS

I.	PREPARING THE SIMULATION MODEL.....	88
A.	GENERAL REMARKS.....	88
B.	LOGIN PROCEDURE.....	89
C.	REQUESTING TEMPORARY DISK SPACE.....	89
D.	READING THE PROGRAMS.....	90
E.	COMPILING THE PROGRAMS.....	90
F.	SAVING THE TEXT FILES.....	91
II.	RUNNING THE SIMULATION MODEL.....	92
A.	STARTING THE SIMULATION.....	92
B.	GENERAL RULE FOR PARAMETER ENTRIES.....	93
C.	ENTERING SYSTEM MODIFICATIONS.....	93
D.	ENTERING RUN PARAMETERS.....	95
E.	ENTERING INITIATOR MODIFICATIONS.....	96
F.	ENTERING TRACE PARAMETERS.....	97
G.	ENTERING RESTART PARAMETERS.....	98
III.	OBTAINING RESULTS.....	100
A.	OBTAINING SIMULATION TRACE AND CORE MAP.....	100
B.	OBTAINING STATISTICAL DATA.....	100

I. PREPARING THE SIMULATION MODEL

A. GENERAL REMARKS

Before working with the simulation model a PROFILE EXEC file should be prepared on P-disk by the user. In the following example the PROFILE used during the simulation is printed.

```
print profile exec
&TYPEOUT OFF
GLOBAL TXTLIB PLILIB
VSET RDYMSG OFF
BLIP *
R;
```

The GLOBAL command is necessary to establish the linkage to certain libraries used at compile and run time. The other commands are optional. The TYPEOUT OFF command suppresses the typing of the PROFILE commands. The command VSET RDYMSG OFF is used to abbreviate the system's error and ready messages. The BLIP command prints an asterisk every two seconds of CPU time used. With means of that the user can see the duration of the different program parts.

Once the PROFILE EXEC file is prepared on the P-disk it is executed automatically by the system whenever the user logs into CP/CMS and executes the first CMS command.

B. LOGIN PROCEDURE

The simulation model needs about 500 K main memory for the program itself and for I/O buffers. This amount of virtual storage must be requested at log-in time.

```
login 0860p10 500k
ENTER PASSWORD:
```

```
*****
ENTER 4-DIGIT PROJECT NUMBER FOLLOWED BY ...
0748cr62
FILES:- 02 RDR, NO PRT, NO PUN
READY AT 10.12.16 ON 05/09/77
CMS VERSION 3.2
```

C. REQUESTING TEMPORARY DISK SPACE

With the following sequence of commands the user will get 7 additional cylinders of temporary space on a B-disk.

```
cp define t2314 192 7
R;

login 192 b
IOERR R 0794 NRF-MADDMK ADDRESS: 192
** B (192) DEVICE ERROR **
E(00001)

format b all
** "FORMAT B" WILL ERASE ALL YOUR B-DISK (192) FILES **
** DO YOU WISH TO CONTINUE? ENTER "YES" OR "NO":
yes
ENTER 6-BYTE LABEL (IF WANTED), OR NULL LINE (IF NOT):
FORMATTING B-DISK (2314)...
B (192): 007 CYL
R;

release 192 b
R;

login 192 p
192 REPLACES P (191)
R;

login 191 b,p
B (191) R/O
R;
```

D. READING THE PROGRAMS

It is assumed that the simulation program (SIM) and the supporting statistical evaluation program (STA) are available as card decks and are prepared for an OFFLINE READ.

```
o read sim pli
R;

o read sta pli
R;
```

E. COMPILING THE PROGRAMS

With the following commands the programs will be compiled. The option FE will suppress compiler warnings.

```
pli sim ( fe )
*****E(00004)

pli sta ( fe )
*****E(00004)

list
FILENAME FILETYPE MODE NO.REC. DATE
SIM      PLI      P1      177    5/09
STA      PLI      P1       40    5/09
SIM      LISTING  P1     360    5/09
SIM      TEXT     P1     177    5/09
STA      LISTING  P1      88    5/09
STA      TEXT     P1      44    5/09
R;

stat
P (192): 6 FILES; 901 REC IN USE, 147 LEFT (OF 1048),
R;                                     86% FULL (7 CYL)
```

With an OFFLINE PRINT command the user can get now a compiler listing. If a listing is not wanted at all the NOPRINT (NP) option should be used to speed up the compilation. [Example : pli sim (fe np)].

Depending upon the current number of users logged into CP/CMS and upon the used compiler options the compilation of SIM will take between 7 and 30 minutes, the compilation of STA between 3 and 15 minutes wall clock time.

F. SAVING THE TEXT FILES

With the following commands the TEXT files (i.e. the compiled version of the programs) are transferred to the p-disk and the temporary disk space is released:

```

cp xfer d to 0860p
R;

o punchcc sim text
** CARDS XFERED BY 0860P10 **
** CARDS XFERED TO 0860P **
R;

o punchcc sta text
** CARDS XFERED BY 0860P10 **
** CARDS XFERED TO 0860P **
R;

cp xfer d off
R;

release 191 b
R;

login 191 p
191 REPLACES P (192)
R;

o read sim text
R;

o read sta text
R;

list
FILENAME FILETYPE MODE NO.REC. DATE
PROFILE EXEC P1 1 5/09
SIM TEXT P1 177 5/09
STA TEXT P1 44 5/09
R;

stat
P (191): 3 FILES; 230 REC IN USE, 66 LEFT (OF 296),
R; 78% FULL (2 CYL)

```

II. RUNNING THE SIMULATION MODEL

A. STARTING THE SIMULATION

When the user has logged into CP/CMS with 500 K and a SIM TEXT file is on his P-disk he can start the simulation program for an interactive run. To be sure to have enough disk space the user should again use temporary space. If the B-disk is already formatted from the previous compilation then exactly the following commands can be used. Otherwise the user must request and format the B-disk as described earlier.

```
login 192 p
192 REPLACES P (191)
R;

login 191 b,p
B (191) R/O
R;

erase * listing
R;

filedef sysin con blksize 80
R;

filedef sysprint con
R;

$ sim
*EXECUTION BEGINS...
```

With this sequence of commands program messages are directed to the user's console and he can enter the simulation parameters from there. However, the statistical output is written to separate files on the disk and must be processed later.

B. GENERAL RULE FOR PARAMETER ENTRIES

When entering parameters to the model one important rule applies: EVERY PARAMETER ENTRY MUST BE FOLLOWED BY AT LEAST ONE BLANK CHARACTER.

The blank character separates different entries from each other. The program requires that even after a single parameter or after the last in a sequence of parameters at least one blank character must be entered before hitting 'carriage return'. Otherwise the program will wait until the required blank is entered.

C. ENTERING SYSTEM MODIFICATIONS

The user is asked if he wants to modify the set-up of system parameters in the simulation model.

If his answer is 0 (or any other number except 1) no modification is wanted and the program goes on.

If his answer is 1 then the following parameters can be entered:

- * input spool capacity (in number of cards)
- * public direct access space (in number of records)
- * core - high address (in K)
- * core - low address (in K)
- * number of disk drives
- * number of tape drives

The correct input format is:

```
mod.spool=nn
mod.space=nn
mod.core_h=nn
mod.core_l=nn
mod.disks=nn
mod.tapes=nn
```

In this format nn represents the appropriate number the user wants to enter. This number is not checked for validity. Entering invalid numbers (negative numbers, or core_l > core_h) will cause unpredictable program behavior and erroneous results.

One or more of the modification parameters can be entered in any order. Duplicate entries are allowed, then the last entry counts.

```
SYSTEM MODIFICATIONS? (1=YES, 0=NO)
1
```

```
ENTER:
mod.tapes=1
mod.disks=0
mod.core_h=640
mod.core_l=140
```

To end the modification mode the user must enter a NULL line (carriage return only).

D. ENTERING RUN PARAMETERS

The user must enter the number of jobs to be generated and the time interval for the next simulation run. If an entry is invalid he is asked to retry. He then has the opportunity to enter modifications to the input job stream. If his answer is the number 0 (or any other number except 1) no modification is wanted and the program goes on.

If his answer is 1 the user can change the value of any global variable and of any variable within the procedure GENERATE_JOBS. To obtain the variable names the user is referred to the program listing. An example how to change the job arrival rate (variable name: sim_parameters.alpha) is given below. Further details about format and restrictions of data-directed input without data list, as it is used in the simulation program, are given in: IBM PL/I(F) Language Reference Manual, chap. 9, paragraph: Data-Directed Data Specifications. To end the modification mode the user must enter a NULL line (carriage return only).

```
RUN PARAMETERS:
TIME (0 < SEC. < 604801) :
3600
JOBS (0 < NUMBER < 1001) :
100
JOB STREAM MODIFICATIONS? (1=YES, 0=NO)
1
ENTER:
sim_parameters.alpha=1.1
```

When these parameters are entered the program will generate the requested number of jobs. Depending upon this number it will take between a few seconds and several minutes of wall clock time.

AD-A042 300

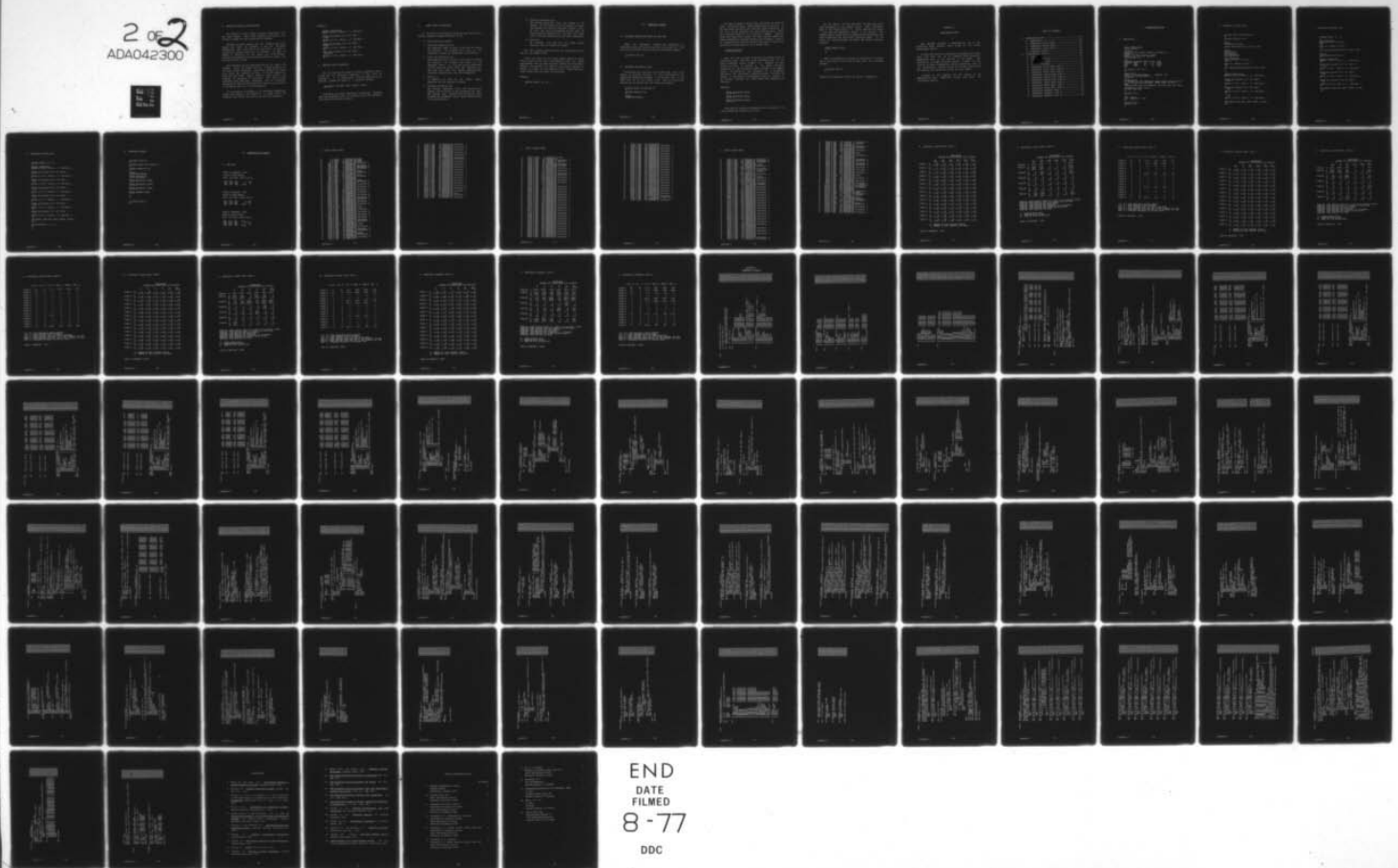
NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF
A SIMULATION MODEL FOR THE STUDY OF JOB SCHEDULING POLICY.(U)
JUN 77 E FIEGL

F/G 9/2

UNCLASSIFIED

NL

2 of 2
ADA042300



E. ENTERING INITIATOR MODIFICATIONS

All entries to start, modify, and stop Initiators have the same format. Each entry consists of two parts: the Initiator number and the associated job classes.

Possible Initiator numbers are 1-15. Entering any other number will cause the program to end the Initiator modification mode and to go to the next step. To start an Initiator an unused number must be entered. To stop or modify an Initiator its old number has to be entered. Multiple modifications of the same Initiator are valid; the last entry will be used.

Each Initiator can be associated with up to eight job classes. Valid job classes are A-O. A blank character in the input stream is skipped. Entering an invalid job class will cause the Initiator to stop. It is good practice to enter the word 'STOP' (which contains three invalid job classes) to terminate an Initiator. Any input string containing more than eight characters will be truncated. Assigning the same job class more than once to one Initiator is illogical, but it is a valid operation.

If an Initiator is stopped, it is terminated immediately except after restart 4 and restart 6. In these cases the Initiator will finish the current job and then terminate.

Examples:

```
MODIFY INITIATORS:
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
1
ENTER JOB CLASSES 'A-O' OR 'STOP' :
'abc'
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
2
ENTER JOB CLASSES 'A-O' OR 'STOP' :
'stop'
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
15
ENTER JOB CLASSES 'A-O' OR 'STOP' :
'oabcefk'
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
0
```

F. ENTERING TRACE PARAMETERS

Now the user has the choice to select simulation trace, a core map, and statistics gathering in that order by entering a 1 as shown below. If these options are not desired, a number not equal to 1 should be entered.

```
SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :
1 1 1
```

Thereafter the actual simulation is executed. Depending upon the parameters used, the simulation will take between a few seconds and several minutes.

G. ENTERING RESTART PARAMETERS

At the end of one simulation step the user must enter a restart parameter ranging from zero to six:

0 - Stop simulation program

1 - Start new simulation run.

The simulation time is reset to zero and the queues are cleared. The program will start at the beginning allowing all parameters to be entered.

2 - Start new simulation run.

The simulation time is reset to zero and the queues are cleared. The program will start at the beginning again. It will use the run parameters and the input job stream from the previous run. The user can enter only system modifications, Initiator modifications, and trace parameters.

3 - Same as 1.

In addition the seed for the random number generator will be changed at random.

4 - Continue simulation run.

The current simulation time, the status of the queues, and the status of the Initiators are kept. The program will continue from the current point allowing all parameters except system modifications to be entered again.

5 - Continue simulation run.

The current simulation time, the status of the queues, and the status of the Initiators are kept. The program will continue from the current point. It will use the run parameters and the input job stream from the previous run. The user can enter only Initiator modifications and trace parameters.

6 - Same as 4.

In addition the seed for the random number generator will be changed at random.

The job classes associated with the Initiators will be kept in all restart cases.

Since the seed for the random number generator starts with a fixed initial value, the simulation runs are always reproducible when the same parameter entries are used again. However, with restart 3 or restart 6 the seed is changed at random by means of the computer clock. These runs use a truly random seed and cannot be reproduced.

Example:

RESTART PARAM. (0 - 6) :
4

III. OBTAINING RESULTS

A. OBTAINING SIMULATION TRACE AND CORE MAP

After the simulation program has terminated the simulation trace and the core map are on file FILE P1. This file can be printed directly using the following command:

```
o printcc file p1
R;
```

B. OBTAINING STATISTICAL DATA

Statistical data are kept on file FILE STAT. This file cannot be printed directly; it must be processed by the supporting statistical program. It is assumed that a STA TEXT file is on the user's disk. Then the statistical program can be started with the following commands:

```
filedef sysin con blksize 80
R;
filedef sysprint con
R;
$ sta
EXECUTION BEGINS...
```

The user is asked to enter those Initiators for which he wants statistical data. SEVEN numbers must be entered. If the user wants statistics for less than seven Initiators, he can fill up the parameter list with trailing zeros. There is no validity check on the Initiator numbers. Trailing zeros will cause blank output; any other invalid number will cause unpredictable program behavior and erroneous results. If statistics for more than seven Initiators are wanted the statistical program must be run a second time.

```
ENTER INITIATORS:  
1 2 15 0 0 0 0
```

Next the user is asked to specify the desired types of statistical data. The first three parameters refer to the three different output formats 1-3. Examples of these formats are given in Appendix B. The fourth parameter indicates if the statistics should be added to the summary output. The purpose of this parameter is to accumulate statistics for continuous simulation steps. Entering the number 1 will cause the appropriate format or summary to be written, entering any other number will suppress this function.

Examples:

```
ENTER STATISTICS TYPES:  
1 1 1 0  
ENTER STATISTICS TYPES:  
1 1 1 1  
ENTER STATISTICS TYPES:  
0 0 0 1
```

The entry of statistics parameters will be repeated for each statistical record on the file.

If the entire file has been read, the user must enter three parameters for the summary report. These parameters refer to the output formats 1-3. Again, entering the number 1 will cause the appropriate format to be written. The summary report should only be requested when statistics have been accumulated from continuous simulation steps. If the steps are non-continuous, the output may contain incorrect results.

```
ENTER SUMMARY TYPES:  
1 1 1
```

```
R;
```

After the statistical program has terminated the results are on file FILE F1 and can be printed using the following command:

```
o printcc file f1  
R;
```

Examples of simulation results are given in Appendix B.

APPENDIX B

DEMONSTRATION RUN

This Appendix contains a demonstration run of the simulation model executed under CP/CMS at the Naval Postgraduate School.

A three hour period is simulated and snapshots are taken after each hour. The use of different input parameters demonstrates most of the features of the model. The restriction in system resources (tapes = 1, disks = 0, core = 500 K) is used to demonstrate certain conditions such as job cancellations because of lack of devices, jobs waiting for devices, and Initiators waiting for main memory.

Included in this Appendix are the results of the demonstration run: simulation traces, core maps, and statistical reports.

TABLE OF CONTENTS

I.	DEMONSTRATION RUN.....	105
A.	PREPARATION.....	105
B.	SIMULATION (FIRST HOUR).....	106
C.	SIMULATION (SECOND HOUR).....	107
D.	SIMULATION (THIRD HOUR).....	108
E.	OBTAINING RESULTS.....	109
II.	DEMONSTRATION RESULTS.....	110
A.	CORE MAPS.....	110
B.	TRACE (FIRST HOUR).....	111
C.	TRACE (SECOND HOUR).....	113
D.	TRACE (THIRD HOUR).....	115
E.	STATISTICS (FIRST HOUR, FORM 1).....	117
F.	STATISTICS (FIRST HOUR, FORM 2).....	118
G.	STATISTICS (FIRST HOUR, FORM 3).....	119
H.	STATISTICS (SECOND HOUR, FORM 1).....	120
I.	STATISTICS (SECOND HOUR, FORM 2).....	121
J.	STATISTICS (SECOND HOUR, FORM 3).....	122
K.	STATISTICS (THIRD HOUR, FORM 1).....	123
L.	STATISTICS (THIRD HOUR, FORM 2).....	124
M.	STATISTICS (THIRD HOUR, FORM 3).....	125
N.	STATISTICS (SUMMARY, FORM 1).....	126
O.	STATISTICS (SUMMARY, FORM 2).....	127
P.	STATISTICS (SUMMARY, FORM 3).....	128

I. DEMONSTRATION RUN

A. PREPARATION

```
login 0860p10 500k
ENTER PASSWORD:
```

```
*****
ENTER 4-DIGIT PROJECT NUMBER FOLLOWED BY ...
0748cr62
FILES:- 02 RDR, NO PRT, NO PUN
READY AT 10.12.16 ON 05/09/77
CMS VERSION 3.2
```

```
list
FILENAME FILETYPE MODE NO.REC. DATE
PROFILE EXEC P1 1 5/09
SIM TEXT P1 177 5/09
STA TEXT P1 44 5/09
R;
```

```
cp define t2314 192 7
R;
```

```
login 192 b
IOERR R 0794 NRF-MADDMK ADDRESS: 192
** B (192) DEVICE ERROR **
E(00001)
```

```
format b all
** "FORMAT B" WILL ERASE ALL YOUR B-DISK (192) FILES **
** DO YOU WISH TO CONTINUE? ENTER "YES" OR "NO":
yes
ENTER 6-BYTE LABEL (IF WANTED), OR NULL LINE (IF NOT):
```

```
FORMATTING B-DISK (2314)...
B (192): 007 CYL
R;
```

```
release 192 b
R;
```

```
login 192 p
192 REPLACES P (191)
R;
```

```
login 191 b,p
B (191) R/O
R;
```

B. SIMULATION (FIRST HOUR)

```
filedef sysin con blksize 80
R;

filedef sysprint con
R;

$ sim
*EXECUTION BEGINS...

SYSTEM MODIFICATIONS? (1=YES, 0=NO)
1

ENTER:
mod.tapes=1
mod.disks=0
mod.core_h=640
mod.core_l=140

RUN PARAMETERS:
TIME (0 < SEC. < 604801) :
3600

JOBS (0 < NUMBER < 1001) :
100

JOB STREAM MODIFICATIONS? (1=YES, 0=NO)
0

**

MODIFY INITIATORS:
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
1

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' abc'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
2

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' oef'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
15

' koa'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
0

SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :
1 1 1
**
```

C. SIMULATION (SECOND HOUR)

```
RESTART PARAM. (0 - 6) :  
4  
RUN PARAMETERS:  
TIME (0 < SEC. < 604801) :  
3600  
JOBS (0 < NUMBER < 1001) :  
100  
JOB STREAM MODIFICATIONS? (1=YES, 0=NO)  
1  
ENTER:  
sim_parameters.alpha=1.1  
**  
MODIFY INITIATORS:  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
15  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' stop'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
3  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' kef'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
2  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' oa'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
0  
SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :  
1 1 1  
**
```


D. SIMULATION (THIRD HOUR)

```
RESTART PARAM. (0 - 6) :  
5  
MODIFY INITIATORS:  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
1  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' o'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
2  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' a'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
3  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' c'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
4  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' b'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
5  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' ef'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
15  
ENTER JOB CLASSES 'A-O' OR 'STOP' :  
' k'  
ENTER N (1-15 = INIT.# , 0 = END MOD.) :  
0  
SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :  
1 1 1  
**  
RESTART PARAM. (0 - 6) :  
0  
R;
```

E. OBTAINING RESULTS

```
o printcc file p1
R;
filedef sysin con blksize 80
R;
filedef sysprint con
R;
$ sta
EXECUTION BEGINS...
ENTER INITIATORS:
1 2 3 4 5 15 0
ENTER STATISTICS TYPES:
1 1 1 1
ENTER STATISTICS TYPES:
1 1 1 1
ENTER STATISTICS TYPES:
1 1 1 1
ENTER SUMMARY TYPES:
1 1 1
R;

o printcc file f1
R;
```

II. DEMONSTRATION RESULTS

A. CORE MAPS

TIME OF SNAPSHOT: 3600

USAGE OF MAIN MEMORY

HIGH	LOW	CORE	INIT.#	JOB #
640	540	100	1	65
540	390	150	2	80
390	140	250	free	

TIME OF SNAPSHOT: 7200

USAGE OF MAIN MEMORY

HIGH	LOW	CORE	INIT.#	JOB #
640	540	100	2	183
540	440	100	1	154
440	140	300	free	

TIME OF SNAPSHOT: 10800

USAGE OF MAIN MEMORY

HIGH	LOW	CORE	INIT.#	JOB #
640	516	124	free	
516	416	100	1	183
416	236	180	4	31
236	140	96	free	

B. TRACE (FIRST HOUR)

*	0	*	INIT.	1	WAITING FOR WORK	
*	0	*	INIT.	2	WAITING FOR WORK	
*	0	*	INIT.	15	WAITING FOR WORK	
*	195	*	JOB	1	STARTED BY INITIATOR	1
*	282	*	JOB	1	TERMINATED	
*	282	*	JOB	2	STARTED BY INITIATOR	1
*	339	*	JOB	3	STARTED BY INITIATOR	15
*	432	*	JOB	7	STARTED BY INITIATOR	2
*	491	*	JOB	7	TERMINATED	
*	491	*	INIT.	2	WAITING FOR WORK	
*	655	*	JOB	13	STARTED BY INITIATOR	2
*	744	*	JOB	13	TERMINATED	
*	744	*	INIT.	2	WAITING FOR WORK	
*	920	*	JOB	17	STARTED BY INITIATOR	2
*	996	*	JOB	17	TERMINATED	
*	996	*	JOB	3	TERMINATED	
*	996	*	INIT.	2	WAITING FOR WORK	
*	996	*	JOB	4	STARTED BY INITIATOR	15
*	1027	*	JOB	4	TERMINATED	
*	1027	*	JOB	6	STARTED BY INITIATOR	15
*	1062	*	JOB	6	TERMINATED	
*	1062	*	JOB	9	STARTED BY INITIATOR	15
*	1265	*	JOB	2	TERMINATED	
*	1265	*	JOB	10	STARTED BY INITIATOR	1
*	1287	*	JOB	10	TERMINATED	
*	1287	*	JOB	12	STARTED BY INITIATOR	1
*	1333	*	JOB	9	TERMINATED	
*	1333	*	JOB	14	STARTED BY INITIATOR	15
*	1445	*	JOB	14	TERMINATED	
*	1445	*	JOB	21	STARTED BY INITIATOR	15
*	1515	*	JOB	27	STARTED BY INITIATOR	2
*	1543	*	JOB	12	TERMINATED	
*	1543	*	JOB	22	STARTED BY INITIATOR	1
*	1574	*	JOB	21	TERMINATED	
*	1574	*	JOB	25	STARTED BY INITIATOR	15
*	1596	*	JOB	27	TERMINATED	
*	1596	*	INIT.	2	WAITING FOR WORK	
*	1793	*	JOB	22	TERMINATED	
*	1793	*	JOB	26	STARTED BY INITIATOR	1
*	1818	*	JOB	25	TERMINATED	
*	1818	*	JOB	28	STARTED BY INITIATOR	15
*	1821	*	JOB	28	TERMINATED	
*	1821	*	JOB	29	STARTED BY INITIATOR	15
*	1851	*	JOB	26	TERMINATED	
*	1851	*	JOB	32	STARTED BY INITIATOR	1
*	1928	*	JOB	32	TERMINATED	
*	1928	*	JOB	33	STARTED BY INITIATOR	1
*	1930	*	JOB	35	STARTED BY INITIATOR	2
*	2038	*	JOB	35	TERMINATED	
*	2038	*	INIT.	2	WAITING FOR WORK	
*	2050	*	JOB	33	TERMINATED	
*	2050	*	JOB	34	STARTED BY INITIATOR	1
*	2133	*	JOB	29	TERMINATED	
*	2133	*	JOB	37	STARTED BY INITIATOR	15
*	2142	*	JOB	38	STARTED BY INITIATOR	2
*	2193	*	JOB	38	TERMINATED	
*	2193	*	INIT.	2	WAITING FOR WORK	
*	2200	*	JOB	40	STARTED BY INITIATOR	2
*	2273	*	JOB	37	TERMINATED	
*	2273	*	JOB	41	STARTED BY INITIATOR	15
*	2369	*	JOB	34	TERMINATED	

*	2369	* JOB	39	STARTED BY	INITIATOR	1
*	2414	* JOB	41	TERMINATED		
*	2414	* JOB	43	STARTED BY	INITIATOR	15
*	2448	* JOB	39	TERMINATED		
*	2448	* JOB	5	STARTED BY	INITIATOR	1
*	2497	* JOB	43	TERMINATED		
*	2497	* JOB	47	STARTED BY	INITIATOR	15
*	2555	* JOB	47	TERMINATED		
*	2555	* JOB	51	STARTED BY	INITIATOR	15
*	2636	* JOB	40	TERMINATED		
*	2636	* JOB	48	STARTED BY	INITIATOR	2
*	2692	* JOB	48	: ALLOCATE	1 DATA SET	
*	2739	* JOB	51	TERMINATED		
*	2739	* JOB	50	STARTED BY	INITIATOR	15
*	2879	* JOB	5	TERMINATED		
*	2879	* JOB	54	STARTED BY	INITIATOR	1
*	3090	* JOB	48	TERMINATED		
*	3090	* JOB	66	STARTED BY	INITIATOR	2
*	3141	* JOB	66	TERMINATED		
*	3141	* JOB	67	STARTED BY	INITIATOR	2
*	3145	* JOB	54	TERMINATED		
*	3145	* JOB	55	STARTED BY	INITIATOR	1
*	3170	* JOB	55	TERMINATED		
*	3170	* JOB	56	STARTED BY	INITIATOR	1
*	3227	* JOB	67	TERMINATED		
*	3227	* JOB	68	STARTED BY	INITIATOR	2
*	3233	* JOB	56	TERMINATED		
*	3233	* JOB	61	STARTED BY	INITIATOR	1
*	3292	* JOB	61	TERMINATED		
*	3292	* JOB	64	STARTED BY	INITIATOR	1
*	3327	* JOB	50	TERMINATED		
*	3327	* JOB	62	STARTED BY	INITIATOR	15
*	3346	* JOB	62	TERMINATED		
*	3346	* JOB	63	STARTED BY	INITIATOR	15
*	3408	* JOB	68	TERMINATED		
*	3408	* JOB	49	STARTED BY	INITIATOR	2
*	3431	* JOB	49	TERMINATED		
*	3431	* INIT.	2	WAITING FOR	WORK	
*	3499	* JOB	80	STARTED BY	INITIATOR	2
*	3564	* JOB	64	TERMINATED		
*	3564	* JOB	65	STARTED BY	INITIATOR	1

C. TRACE (SECOND HOUR)

```

* 3600 * INIT.      3 WAITING FOR WORK
* 3673 * JOB        80 TERMINATED
* 3673 * JOB        69 STARTED BY INITIATOR 2
* 3719 * JOB        63 : MOUNT 1 TAPE(S)
* 3737 * JOB        65 TERMINATED
* 3737 * JOB        70 STARTED BY INITIATOR 1
* 3856 * JOB        63 ALL VOLUMES MOUNTED
* 3904 * JOB        69 TERMINATED
* 3904 * JOB       101 STARTED BY INITIATOR 2
* 3987 * JOB        70 TERMINATED
* 3987 * JOB        72 STARTED BY INITIATOR 1
* 4008 * JOB       101 TERMINATED
* 4008 * JOB       103 STARTED BY INITIATOR 2
* 4035 * JOB       113 STARTED BY INITIATOR 3
* 4070 * JOB       103 TERMINATED
* 4070 * JOB       109 STARTED BY INITIATOR 2
* 4239 * JOB       109 TERMINATED
* 4239 * JOB       110 STARTED BY INITIATOR 2
* 4264 * JOB       113 TERMINATED
* 4264 * JOB       120 STARTED BY INITIATOR 3
* 4349 * JOB        63 TERMINATED
* 4473 * JOB        72 TERMINATED
* 4473 * JOB        73 STARTED BY INITIATOR 1
* 4536 * JOB        73 TERMINATED
* 4536 * JOB        74 STARTED BY INITIATOR 1
* 4555 * JOB       110 TERMINATED
* 4555 * JOB       111 STARTED BY INITIATOR 2
* 4581 * JOB       111 TERMINATED
* 4581 * JOB       118 STARTED BY INITIATOR 2
* 4653 * JOB        74 TERMINATED
* 4653 * JOB        76 STARTED BY INITIATOR 1
* 4672 * JOB       118 TERMINATED
* 4672 * JOB       119 STARTED BY INITIATOR 2
* 4781 * JOB        76 TERMINATED
* 4781 * JOB       105 STARTED BY INITIATOR 1
* 4819 * JOB       119 TERMINATED
* 4819 * JOB       123 STARTED BY INITIATOR 2
* 4924 * JOB       105 TERMINATED
* 4924 * JOB       107 STARTED BY INITIATOR 1
* 5001 * JOB       123 TERMINATED
* 5001 * JOB       125 STARTED BY INITIATOR 2
* 5116 * JOB       107 TERMINATED
* 5116 * JOB       114 STARTED BY INITIATOR 1
* 5138 * JOB       125 TERMINATED
* 5138 * JOB       129 STARTED BY INITIATOR 2
* 5186 * JOB       129 TERMINATED
* 5186 * JOB       136 STARTED BY INITIATOR 2
* 5376 * JOB       136 TERMINATED
* 5376 * JOB       137 STARTED BY INITIATOR 2
* 5379 * JOB       137 TERMINATED
* 5379 * JOB       139 STARTED BY INITIATOR 2
* 5387 * JOB       139 TERMINATED
* 5387 * JOB       142 STARTED BY INITIATOR 2
* 5494 * JOB       142 TERMINATED
* 5494 * JOB       144 STARTED BY INITIATOR 2
* 5510 * JOB       114 TERMINATED
* 5510 * JOB       115 STARTED BY INITIATOR 1
* 5585 * JOB       115 TERMINATED
* 5585 * JOB       116 STARTED BY INITIATOR 1
* 5589 * JOB       144 TERMINATED
* 5589 * JOB       121 STARTED BY INITIATOR 2

```

*	5733	*	JOB	116	TERMINATED		
*	5733	*	JOB	122	STARTED BY	INITIATOR	1
*	5773	*	JOB	122	TERMINATED		
*	5773	*	JOB	126	STARTED BY	INITIATOR	1
*	5924	*	JOB	126	TERMINATED		
*	5924	*	JOB	128	STARTED BY	INITIATOR	1
*	5942	*	JOB	128	TERMINATED		
*	5942	*	JOB	133	STARTED BY	INITIATOR	1
*	6078	*	JOB	133	TERMINATED		
*	6078	*	JOB	138	STARTED BY	INITIATOR	1
*	6134	*	JOB	121	TERMINATED		
*	6134	*	JOB	156	STARTED BY	INITIATOR	2
*	6301	*	JOB	156	TERMINATED		
*	6301	*	JOB	157	STARTED BY	INITIATOR	2
*	6339	*	JOB	157	TERMINATED		
*	6339	*	JOB	141	STARTED BY	INITIATOR	2
*	6353	*	JOB	138	TERMINATED		
*	6353	*	JOB	143	STARTED BY	INITIATOR	1
*	6366	*	JOB	141	TERMINATED		
*	6366	*	JOB	145	STARTED BY	INITIATOR	2
*	6372	*	JOB	143	TERMINATED		
*	6372	*	JOB	146	STARTED BY	INITIATOR	1
*	6609	*	JOB	145	TERMINATED		
*	6609	*	JOB	171	STARTED BY	INITIATOR	2
*	6735	*	JOB	171	TERMINATED		
*	6735	*	JOB	177	STARTED BY	INITIATOR	2
*	6749	*	JOB	146	TERMINATED		
*	6749	*	JOB	147	STARTED BY	INITIATOR	1
*	6772	*	JOB	177	TERMINATED		
*	6772	*	JOB	149	STARTED BY	INITIATOR	2
*	6918	*	JOB	147	TERMINATED		
*	6918	*	JOB	151	STARTED BY	INITIATOR	1
*	6965	*	JOB	149	TERMINATED		
*	6965	*	JOB	181	STARTED BY	INITIATOR	2
*	6980	*	JOB	181	TERMINATED		
*	6980	*	JOB	152	STARTED BY	INITIATOR	2
*	7009	*	JOB	151	TERMINATED		
*	7009	*	JOB	153	STARTED BY	INITIATOR	1
*	7097	*	JOB	152	TERMINATED		
*	7097	*	JOB	183	STARTED BY	INITIATOR	2
*	7156	*	JOB	153	: ALLOCATE	1 DATA SET	
*	7198	*	JOB	153	TERMINATED		
*	7198	*	JOB	154	STARTED BY	INITIATOR	1

D. TRACE (THIRD HOUR)

*	7200	* JOB	15	STARTED BY INITIATOR	4
*	7200	* JOB	117	STARTED BY INITIATOR	5
*	7200	* JOB	130	STARTED BY INITIATOR	15
*	7268	* JOB	154	TERMINATED	
*	7268	* INIT.	1	WAITING FOR WORK	
*	7302	* JOB	101	STARTED BY INITIATOR	1
*	7336	* JOB	183	TERMINATED	
*	7336	* JOB	158	STARTED BY INITIATOR	2
*	7336	* JOB	158	: ALLOCATE 1 DATA SET	
*	7370	* JOB	15	: MOUNT 1 TAPE(S)	
*	7406	* JOB	101	TERMINATED	
*	7406	* JOB	103	STARTED BY INITIATOR	1
*	7468	* JOB	103	TERMINATED	
*	7468	* INIT.	1	WAITING FOR WORK	
*	7484	* JOB	109	STARTED BY INITIATOR	1
*	7653	* JOB	109	TERMINATED	
*	7653	* JOB	110	STARTED BY INITIATOR	1
*	7768	* JOB	158	TERMINATED	
*	7768	* JOB	159	STARTED BY INITIATOR	2
*	7857	* JOB	15	ALL VOLUMES MOUNTED	
*	7882	* JOB	15	TERMINATED	
*	7882	* JOB	16	STARTED BY INITIATOR	4
*	7969	* JOB	110	TERMINATED	
*	7969	* JOB	111	STARTED BY INITIATOR	1
*	7995	* JOB	111	TERMINATED	
*	7995	* JOB	118	STARTED BY INITIATOR	1
*	8031	* JOB	159	TERMINATED	
*	8031	* JOB	160	STARTED BY INITIATOR	2
*	8086	* JOB	118	TERMINATED	
*	8086	* JOB	119	STARTED BY INITIATOR	1
*	8170	* JOB	160	TERMINATED	
*	8170	* JOB	167	STARTED BY INITIATOR	2
*	8182	* JOB	167	TERMINATED	
*	8182	* JOB	170	STARTED BY INITIATOR	2
*	8233	* JOB	119	TERMINATED	
*	8233	* JOB	123	STARTED BY INITIATOR	1
*	8251	* JOB	170	TERMINATED	
*	8251	* JOB	174	STARTED BY INITIATOR	2
*	8400	* JOB	174	TERMINATED	
*	8400	* JOB	179	STARTED BY INITIATOR	2
*	8415	* JOB	123	TERMINATED	
*	8415	* JOB	125	STARTED BY INITIATOR	1
*	8552	* JOB	125	TERMINATED	
*	8552	* JOB	129	STARTED BY INITIATOR	1
*	8600	* JOB	129	TERMINATED	
*	8600	* JOB	136	STARTED BY INITIATOR	1
*	8769	* JOB	16	TERMINATED	
*	8769	* JOB	18	STARTED BY INITIATOR	4
*	8784	* JOB	179	TERMINATED	
*	8784	* JOB	180	STARTED BY INITIATOR	2
*	8790	* JOB	136	TERMINATED	
*	8797	* JOB	180	TERMINATED	
*	8797	* JOB	137	STARTED BY INITIATOR	1
*	8800	* JOB	137	TERMINATED	
*	8800	* JOB	105	STARTED BY INITIATOR	2
*	8876	* JOB	18	TERMINATED	
*	8876	* JOB	139	STARTED BY INITIATOR	1
*	8884	* JOB	139	TERMINATED	
*	8928	* JOB	117	TERMINATED	
*	8928	* JOB	142	STARTED BY INITIATOR	1
*	8928	* JOB	19	STARTED BY INITIATOR	4

*	8928	* JOB	176	STARTED BY INITIATOR	5
*	8958	* JOB	19	: MOUNT 1 TAPE(S)	
*	8989	* JOB	105	TERMINATED	
*	8989	* JOB	107	STARTED BY INITIATOR	2
*	9027	* JOB	19	ALL VOLUMES MOUNTED	
*	9035	* JOB	142	TERMINATED	
*	9035	* JOB	144	STARTED BY INITIATOR	1
*	9130	* JOB	144	TERMINATED	
*	9130	* INIT.	1	WAITING FOR WORK	
*	9147	* JOB	19	TERMINATED	
*	9147	* JOB	20	STARTED BY INITIATOR	4
*	9181	* JOB	107	TERMINATED	
*	9191	* JOB	176	TERMINATED	
*	9191	* JOB	114	STARTED BY INITIATOR	2
*	9191	* JOB	117	STARTED BY INITIATOR	5
*	9585	* JOB	114	TERMINATED	
*	9585	* JOB	115	STARTED BY INITIATOR	2
*	9605	* JOB	156	STARTED BY INITIATOR	1
*	9723	* JOB	117	TERMINATED	
*	9723	* JOB	178	STARTED BY INITIATOR	5
*	9772	* JOB	156	TERMINATED	
*	9772	* JOB	157	STARTED BY INITIATOR	1
*	9798	* JOB	115	TERMINATED	
*	9798	* JOB	116	STARTED BY INITIATOR	2
*	9810	* JOB	157	TERMINATED	
*	9810	* INIT.	1	WAITING FOR WORK	
*	9901	* JOB	20	TERMINATED	
*	9901	* JOB	23	STARTED BY INITIATOR	4
*	9946	* JOB	116	TERMINATED	
*	9946	* JOB	121	STARTED BY INITIATOR	2
*	9986	* JOB	23	TERMINATED	
*	9986	* JOB	30	STARTED BY INITIATOR	4
*	10149	* JOB	178	WAITING FOR 1 DISK(S)	
*	10186	* JOB	171	STARTED BY INITIATOR	1
*	10278	* JOB	178	CANCELLED BY OPERATOR	
*	10278	* JOB	178	TERMINATED	
*	10312	* JOB	171	TERMINATED	
*	10312	* JOB	176	STARTED BY INITIATOR	5
*	10312	* INIT.	1	WAITING FOR WORK	
*	10334	* JOB	177	STARTED BY INITIATOR	1
*	10364	* JOB	30	: MOUNT 1 TAPE(S)	
*	10371	* JOB	177	TERMINATED	
*	10371	* INIT.	1	WAITING FOR WORK	
*	10500	* JOB	30	ALL VOLUMES MOUNTED	
*	10511	* JOB	181	STARTED BY INITIATOR	1
*	10526	* JOB	181	TERMINATED	
*	10526	* INIT.	1	WAITING FOR WORK	
*	10630	* JOB	183	STARTED BY INITIATOR	1
*	10632	* JOB	30	TERMINATED	
*	10632	* JOB	31	STARTED BY INITIATOR	4
*	10750	* JOB	121	TERMINATED	
*	10750	* JOB	122	STARTED BY INITIATOR	2
*	10790	* JOB	122	TERMINATED	
*	10790	* JOB	126	STARTED BY INITIATOR	2

E. STATISTICS (FIRST HOUR, FORM 1)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 ABC	2 OEF	3 STOP	4 STOP	5 STOP	15 STOP
CLASS A	1:	15	0	0	0	0	0
	2:	0.250	0.000	0.000	0.000	0.000	0.000
CLASS B	1:	1	0	0	0	0	0
	2:	0.017	0.000	0.000	0.000	0.000	0.000
CLASS C	1:	1	0	0	0	0	0
	2:	0.017	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS F	1:	0	2	0	0	0	0
	2:	0.000	0.033	0.000	0.000	0.000	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	0	11	0	0	0	0
	2:	0.000	0.183	0.000	0.000	0.000	0.000

1: NUMBER OF JOBS STARTED (TOTAL)
2: NUMBER OF JOBS STARTED (PER MIN.)

TIME OF SNAPSHOT: 3600

F. STATISTICS (FIRST HOUR, FORM 2)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 ABC	2 OEF	3 STOP	4 STOP	5 STOP	15 STOP
TIME_AC	1:	3600	3600	0	0	0	3600
TIME_SJ	1:	3405	1664	0	0	0	2968
	2:	953	436	0	0	0	405
	3:	94.6	46.2	0.0	0.0	0.0	82.4
TIME_WM	1:	0	125	0	0	0	293
	2:	0	125	0	0	0	254
	3:	0.0	3.5	0.0	0.0	0.0	8.1
TIME_WD	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WV	1:	0	7	0	0	0	0
	2:	0	7	0	0	0	0
	3:	0.0	0.2	0.0	0.0	0.0	0.0
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	195	1804	0	0	0	339
	2:	195	519	0	0	0	339
	3:	5.4	50.1	0.0	0.0	0.0	9.4

TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES)
TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
TIME_WM: TIME WAITING FOR MAIN MEMORY
TIME_WD: TIME WAITING FOR DEVICE(S)
TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
2: MAXIMUM TIME IN SEC.
3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 3600

G. STATISTICS (FIRST HOUR, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	32	26	81.3	748	333	235
CLASS B	20	1	5.0	2087	2087	--
CLASS C	12	1	8.3	50	50	--
CLASS D	0	0	--	--	--	--
CLASS E	0	0	--	--	--	--
CLASS F	2	2	100.0	1015	653	513
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	2	2	100.0	558	550	11
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	15	15	100.0	212	58	78

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
 J.S. # = JOBS STARTED (TOTAL NUMBER)
 J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
 MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
 MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
 DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 3600

H. STATISTICS (SECOND HOUR, FORM 1)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 ABC	2 OA	3 KEF	4 STOP	5 STOP	15 STOP
CLASS A	1:	21	6	0	0	0	0
	2:	0.350	0.100	0.000	0.000	0.000	0.000
CLASS B	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS C	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS F	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	2	0	0	0
	2:	0.000	0.000	0.033	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	0	21	0	0	0	0
	2:	0.000	0.350	0.000	0.000	0.000	0.000
1:		NUMBER OF JOBS STARTED (TOTAL)					
2:		NUMBER OF JOBS STARTED (PER MIN.)					

TIME OF SNAPSHOT: 7200

I. STATISTICS (SECOND HOUR, FORM 2)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 ABC	2 OA	3 KEF	4 STOP	5 STOP	15 STOP
TIME_AC	1:	3600	3600	3600	0	0	749
TIME_SJ	1:	3506	3536	229	0	0	437
	2:	326	439	229	0	0	208
	3:	97.4	98.2	6.4	0.0	0.0	58.3
TIME_WM	1:	86	64	2936	0	0	75
	2:	86	64	2936	0	0	102
	3:	2.4	1.8	81.6	0.0	0.0	23.4
TIME_WD	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WV	1:	8	0	0	0	0	137
	2:	8	0	0	0	0	137
	3:	0.2	0.0	0.0	0.0	0.0	18.3
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	0	0	435	0	0	0
	2:	0	0	435	0	0	0
	3:	0.0	0.0	12.1	0.0	0.0	0.0

TIME_AC: TIME ACTIVE (=TIME SJ + SUM OF ALL WAITING TIMES)
TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
TIME_WM: TIME WAITING FOR MAIN MEMORY
TIME_WD: TIME WAITING FOR DEVICE(S)
TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
2: MAXIMUM TIME IN SEC.
3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 7200

J. STATISTICS (SECOND HOUR, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	35	27	77.1	1479	1224	232
CLASS B	30	0	--	--	--	--
CLASS C	23	0	--	--	--	--
CLASS D	0	0	--	--	--	--
CLASS E	2	0	--	--	--	--
CLASS F	1	0	--	--	--	--
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	8	2	25.0	13	7	9
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	21	21	100.0	651	323	206

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
 J.A. # = JOBS STARTED (TOTAL NUMBER)
 J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
 MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
 MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
 DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 7200

K. STATISTICS (THIRD HOUR, FORM 1)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 O	2 A	3 C	4 B	5 EF	15 STOP
CLASS A	1:	0	16	0	0	0	0
	2:	0.000	0.267	0.000	0.000	0.000	0.000
CLASS B	1:	0	0	0	8	0	0
	2:	0.000	0.000	0.000	0.133	0.000	0.000
CLASS C	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	4	0
	2:	0.000	0.000	0.000	0.000	0.067	0.000
CLASS F	1:	0	0	0	0	1	0
	2:	0.000	0.000	0.000	0.000	0.017	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	21	0	0	0	0	0
	2:	0.350	0.000	0.000	0.000	0.000	0.000
1:		NUMBER OF JOBS STARTED (TOTAL)					
2:		NUMBER OF JOBS STARTED (PER MIN.)					

TIME OF SNAPSHOT: 10800

L. STATISTICS (THIRD HOUR, FORM 2)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 O	2 A	3 C	4 B	5 EF	15 STOP
TIME_AC	1:	3600	3600	3600	3600	3600	3600
TIME_SJ	1:	2306	3120	0	2733	750	9
	2:	316	439	0	462	203	9
	3:	64.1	86.7	0.0	75.9	20.8	0.3
TIME_WM	1:	127	466	3600	175	2721	3591
	2:	76	259	3600	69	1571	3591
	3:	3.5	12.9	100.0	4.9	75.6	99.8
TIME_WD	1:	0	0	0	0	129	0
	2:	0	0	0	0	129	0
	3:	0.0	0.0	0.0	0.0	3.6	0.0
TIME_WV	1:	0	14	0	692	0	0
	2:	0	14	0	487	0	0
	3:	0.0	0.4	0.0	19.2	0.0	0.0
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	1167	0	0	0	0	0
	2:	475	0	0	0	0	0
	3:	32.4	0.0	0.0	0.0	0.0	0.0

TIME_AC: TIME ACTIVE (=TIME SJ + SUM OF ALL WAITING TIMES)
 TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
 TIME_WM: TIME WAITING FOR MAIN MEMORY
 TIME_WD: TIME WAITING FOR DEVICE(S)
 TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
 TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
 TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
 2: MAXIMUM TIME IN SEC.
 3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 10800

M. STATISTICS (THIRD HOUR, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	37	16	43.2	2814	1850	439
CLASS B	41	8	19.5	8884	7830	794
CLASS C	35	0	--	--	--	--
CLASS D	0	0	--	--	--	--
CLASS E	4	4	100.0	3084	1697	1305
CLASS F	2	1	50.0	2962	2962	--
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	14	1	7.1	2711	2711	--
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	21	21	100.0	465	185	167

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
 J.A. # = JOBS STARTED (TOTAL NUMBER)
 J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
 MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
 MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
 DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 10800

N. STATISTICS (SUMMARY, FORM 1)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 O	2 A	3 C	4 B	5 EF	15 STOP
CLASS A	1:	36	22	0	0	0	0
	2:	0.200	0.122	0.000	0.000	0.000	0.000
CLASS B	1:	1	0	0	8	0	0
	2:	0.006	0.000	0.000	0.133	0.000	0.000
CLASS C	1:	1	0	0	0	0	0
	2:	0.006	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	4	0
	2:	0.000	0.000	0.000	0.000	0.067	0.000
CLASS F	1:	0	2	0	0	1	0
	2:	0.000	0.011	0.000	0.000	0.017	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	2	0	0	0
	2:	0.000	0.000	0.017	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	21	32	0	0	0	0
	2:	0.117	0.178	0.000	0.000	0.000	0.000

1: NUMBER OF JOBS STARTED (TOTAL)
2: NUMBER OF JOBS STARTED (PER MIN.)

TIME OF SNAPSHOT: 10800

O. STATISTICS (SUMMARY, FORM 2)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 O	2 A	3 C	4 B	5 EF	15 STOP
TIME_AC	1:	10800	10800	7200	3600	3600	7949
TIME_SJ	1:	9217	8320	229	2733	750	3414
	2:	953	439	229	462	203	405
	3:	85.3	77.0	3.2	75.9	20.8	42.9
TIME_WM	1:	213	655	6536	175	2721	4059
	2:	86	259	3600	69	1571	3591
	3:	2.0	6.1	90.8	4.9	75.6	51.1
TIME_WD	1:	0	0	0	0	129	0
	2:	0	0	0	0	129	0
	3:	0.0	0.0	0.0	0.0	3.6	0.0
TIME_WV	1:	8	21	0	692	0	137
	2:	8	14	0	487	0	137
	3:	0.1	0.2	0.0	19.2	0.0	1.7
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	1362	1804	435	0	0	339
	2:	475	519	435	0	0	339
	3:	12.6	16.7	6.0	0.0	0.0	4.3

TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES)
 TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
 TIME_WM: TIME WAITING FOR MAIN MEMORY
 TIME_WD: TIME WAITING FOR DEVICE(S)
 TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
 TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
 TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
 2: MAXIMUM TIME IN SEC.
 3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 10800

P. STATISTICS (SUMMARY, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	90	69	76.7	2814	1033	665
CLASS B	42	9	21.4	8884	7192	2054
CLASS C	36	1	2.8	50	50	--
CLASS D	0	0	--	--	--	--
CLASS E	4	4	100.0	3084	1697	1305
CLASS F	4	3	75.0	2962	1422	1382
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	18	5	27.8	2711	765	1121
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	57	57	100.0	651	202	194

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
 J.A. # = JOBS STARTED (TOTAL NUMBER)
 J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
 MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
 MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
 DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 10800

COMPUTER PROGRAMS

SIM:

IM000420
 SIM000430
 SIM000440
 SIM000450
 SIM000460
 SIM000470
 SIM000480
 SIM000490
 SIM000500
 SIM000510
 SIM000520
 SIM000530
 SIM000540
 SIM000550
 SIM000560
 SIM000570
 SIM000580
 SIM000590
 SIM000600
 SIM000610
 SIM000620
 SIM000630
 SIM000640
 SIM000650
 SIM000660
 SIM000670
 SIM000680
 SIM000690
 SIM000700
 SIM000710
 SIM000720
 SIM000730
 SIM000740
 SIM000750
 SIM000760
 SIM000770
 SIM000780
 SIM000790
 SIM000800

DCL	1	INPUT STREAM,	LIKE JOB,	
	2	INPUT(5000)	FIXED BIN,	
	2	JOBS(1000)	FIXED BIN,	
	2	STEP_COUNT	FIXED BIN,	
	2	JOB_COUNT	FIXED BIN,	
DCL	1	JOB ARRIVAL	(JOB_PTR),	
	2	DEL ARRIVAL	FIXED BIN(31),	
	2	PRIORITY	FIXED BIN,	
	2	NUMBER	FIXED BIN,	
	2	STEPS	FIXED BIN,	
	2	CLASS	FIXED BIN,	
	2	CARDS	FIXED BIN,	
DCL	1	STEP BASED	(STEP_PTR),	
	2	RUN TIME	FIXED BIN(31),	
	2	NUMBER SIZE	FIXED BIN,	
	2	DISKS	FIXED BIN,	
	2	TAPES	FIXED BIN,	
	2	OP REQUESTS	FIXED BIN,	
	2	D_A_SPACE	FIXED BIN,	
DCL	1	JOB QUEUE,	FIXED BIN INIT (5000),	
	2	SIZE HEAD	FIXED BIN,	
	2	AMOUNT(15)	FIXED BIN,	
	2	HEAD(15)	FIXED BIN,	
	2	LINK(5000),	FIXED BIN,	
	3	JOB LINK	FIXED BIN,	
	3	STEP LINK	FIXED BIN,	
	2	QUEUE(5000)	LIKE JOB;	
DCL	1	TIME	FIXED BIN(31),	
	2	SYSTEM	FIXED BIN(31),	
	2	STOP	FIXED BIN(31),	
	2	READER	FIXED BIN(31),	
	2	INITIATOR(15)	FIXED BIN(31);	

DCL	1	IN(15),	FIXED	BIN,	SIM00820
	2	STATUS	FIXED	BIN,	SIM00830
	2	JOB-NO	FIXED	BIN,	SIM00840
	2	JOB-INDEX	FIXED	BIN,	SIM00850
	2	STEP-INDEX	FIXED	BIN,	SIM00860
	2	STOP-SWITCH	FIXED	BIN,	SIM00870
	2	CLASS(8)	FIXED	BIN,	SIM00880
	2	RE-SOURCES,			SIM00890
	3	D.A. SPACE	FIXED	BIN,	SIM00900
	3	CORE-HIGH	FIXED	BIN,	SIM00910
	3	CORE-LOW	FIXED	BIN,	SIM00920
	3	DISKS	FIXED	BIN,	SIM00930
	3	TAPES	FIXED	BIN,	SIM00940
	3	CARDS	FIXED	BIN,	SIM00950
					SIM00960
DCL	1	STATISTICS,			SIM00970
	2	TIME-LC(15)	FIXED	BIN(31),	SIM00980
	2	TIME-SNAP	FIXED	BIN(31),	SIM00990
	2	MAXIT(15),			SIM01000
	3	JOB-WAITM	FIXED	BIN(31),	SIM01010
	3	MAX-SJ	FIXED	BIN(31),	SIM01020
	3	MAX-MD	FIXED	BIN(31),	SIM01030
	3	MAX-WS	FIXED	BIN(31),	SIM01040
	3	MAX-WV	FIXED	BIN(31),	SIM01050
	3	MAX-WW	FIXED	BIN(31),	SIM01060
	3	NO(15),			SIM01070
	2	TIME-SJ	FIXED	BIN(31),	SIM01080
	3	TIME-WM	FIXED	BIN(31),	SIM01090
	3	TIME-WD	FIXED	BIN(31),	SIM01100
	3	TIME-WV	FIXED	BIN(31),	SIM01110
	3	TIME-WS	FIXED	BIN(31),	SIM01120
	3	TIME-WW	FIXED	BIN(31),	SIM01130
	3	JOB-WAIT	FIXED	BIN(31),	SIM01140
	3	JOB-WAITS	FIXED	BIN(31),	SIM01150
	3	JOB-WAITS	FIXED	BIN,	SIM01160
	3	JOB-START(15)	FIXED	BIN,	SIM01170
	3	IN-QUEUE	FIXED	BIN,	SIM01180
	3	ABEND-DEVICE	FIXED	BIN,	SIM01190
	3	ABEND-D-SET	FIXED	BIN,	SIM01200
	3	ABEND-SPACE	FIXED	BIN,	SIM01210
	3	OP-ANSWER	FIXED	BIN,	SIM01220
	3	OP-TIME	FIXED	BIN,	SIM01230
	3	JOB-IN	FIXED	BIN,	SIM01240
	2	CLASSES(15)	CHAR	(8);	SIM01250


```

GENERATE JOBS: PROCEDURE;
7* GENERATE INPUT JOB STREAM */

DCL SPLIT(15) LABEL
  INIT (
    CLASS_A, CLASS_B, CLASS_C, CLASS_K, CLASS_E,
    CLASS_F, CLASS_K, CLASS_K, CLASS_K, CLASS_K,
    CLASS_K, CLASS_K, CLASS_K, CLASS_K, CLASS_C);

DCL CLASSES(15) INIT(
  .3664, .5832, .7086, .7086, .7200,
  .7310, .7825, .7825, .7825, .7825,
  .7825, .7825, .7825, .7825, 1.00);

DCL DEVICES_1(10) INIT(
  .8887, .9283, .9302, .9321, .9326,
  .9325, .9332, .9337, .9521, 1.00);

DCL DEVICES_2(11) INIT(
  .8570, .8951, .8970, .8988, .8993,
  .8996, .8999, .9004, .9567, .9643,
  1.00);

DCL DISK(11) FIXED BIN INIT( 0,1,2,1,1,2,3,3,0,0,0 );
DCL TAPE(11) FIXED BIN INIT( 0,0,0,1,2,2,1,2,1,2,3 );

DCL REQUESTS(2) INIT ( .9625, 1.0);
DCL RANDOM_X;

REAL: PROCEDURE (TABLE);
DCL TABLE(60), NUMBER, VALUE, X_LOW;
CALL RANDU (SEED_1, SEED_1, RANDOM_X);
NUMBER = 1;
DO WHILE (RANDOM_X > TABLE(NUMBER));
  NUMBER = NUMBER + 1;
END;
IF NUMBER = 1 THEN X_LOW = 0;
ELSE X_LOW = TABLE (NUMBER - 1);
VALUE = NUMBER - 1 + (RANDOM_X - X_LOW) / (TABLE(NUMBER) - X_LOW);
RETURN (VALUE);
END REAL;

```

SIM01280
SIM01290
SIM01300
SIM01310
SIM01320
SIM01330
SIM01340
SIM01350
SIM01360
SIM01370
SIM01380
SIM01390
SIM01400
SIM01410
SIM01420
SIM01430
SIM01440
SIM01450
SIM01460
SIM01470
SIM01480
SIM01490
SIM01500
SIM01510
SIM01520
SIM01530
SIM01540
SIM01550
SIM01560
SIM01570
SIM01580
SIM01590
SIM01600
SIM01610
SIM01620
SIM01630
SIM01640
SIM01650
SIM01660
SIM01670
SIM01680
SIM01690

```

INTEGER:      PROCEDURE (TABLE);
              DCL  TABLE(60), NUMBER;
              CALL RANDU (SEED_1,SEED_1,RANDOM_X);
              NUMBER = 1;
              DO WHILE (RANDOM_X > TABLE(NUMBER));
              NUMBER = NUMBER + 1;
              END;
              RETURN (NUMBER);
              END INTEGER;

SET_DEL_ARRIVAL: PROCEDURE;
              CALL RANDU(SEED_1,SEED_1,RANDOM_X);
              RANDOM_X = - LOG(RANDOM_X) / ALPHA * 60;
              RETURN(RANDOM_X);
              END SET_DEL_ARRIVAL;

/* SET JOB PARAMETERS */
PUT SKIP LIST ('JOB STREAM MODIFICATIONS? (1=YES, 0=NO)');
GET LIST (INDEX);
IF INDEX=1
  THEN
    DO;
    PUT SKIP LIST ('ENTER:');
    PUT SKIP;
    GET DATA;
    END;
  INDEX = 1;
  DO I=1 TO MAX_JOBS;
    JOBS(I)
    JOB_PTR
    JOB_INDEX
    JOB_DEL_ARRIVAL = INDEX; INPUT(INDEX);
    JOB_CLASS = INDEX + 1;
    JOB_PRIORITY = SET_DEL_ARRIVAL + .5;
    JOB_ARRIVAL = INTEGER(CLASSES);
    JOB_NUMBER = 1;
    JOB_NUMBER = JOB_NUMBER + 1;
    GOT0 SPLIT(JOB.CLASS);
  
```

SIM01710
 SIM01720
 SIM01730
 SIM01740
 SIM01750
 SIM01760
 SIM01770
 SIM01780
 SIM01790
 SIM01800
 SIM01810
 SIM01820
 SIM01830
 SIM01840
 SIM01850
 SIM01860
 SIM01870
 SIM01880
 SIM01890
 SIM01900
 SIM01910
 SIM01920
 SIM01930
 SIM01940
 SIM01950
 SIM01960
 SIM01961
 SIM01962
 SIM01963
 SIM01964
 SIM01965
 SIM01966
 SIM01967
 SIM01968
 SIM01969
 SIM01970
 SIM01971
 SIM01980
 SIM01990
 SIM02000
 SIM02010
 SIM02020
 SIM02030
 SIM02040
 SIM02050
 SIM02060
 SIM02070
 SIM02080

```

CLASS_A::
DCL  STEPS_A(11) INIT(
      .4180, .4711, .9780, .9866, .5985,
      .9995, .9995, .9995, .9998, .9999,
      1.00);
DCL  CARDS_A(30) INIT(
      .3385, .5071, .5878, .6565, .7158,
      .7775, .8190, .8497, .8744, .8988,
      .9190, .9297, .9400, .9463, .9518,
      .9547, .9587, .9630, .9655, .9706,
      .9764, .9787, .9827, .9870, .9906,
      .9933, .9960, .9978, .9995, 1.00);
DCL  CORE_A(13) INIT(
      .3074, .3118, .3149, .3228, .8267,
      .8322, .8421, .8483, .8485, .9618,
      .9655, .9704, 1.00);
DCL  TIME_A(44) INIT(
      .0621, .1251, .2230, .3358, .4406,
      .5349, .6139, .6734, .7245, .7682,
      .8049, .8356, .8603, .8813, .8983,
      .9115, .9238, .9334, .9423, .9487,
      .9550, .9601, .9648, .9684, .9711,
      .9743, .9771, .9794, .9815, .9833,
      .9852, .9868, .9884, .9899, .9912,
      .9929, .9941, .9950, .9963, .9975,
      .9980, .9990, .9996, 1.00);

JOB STEPS = INTEGER(STEPS_A);
JOB CARDS = REAL(CARDS_A)*40 + .5;
DO J=1 TO JOB.STEPS;
  STEP PTR = ADDR(INPUT(INDEX));
  INDEX = INDEX + 1;
  STEP.RUN_TIME = REAL(TIME_A) * 10 + .5;
  STEP.NUMBER = J;
  STEP.CORE_SIZE = INTEGER(CORE_A) * 10 + 50;
  STEP.TAPES = 0;
  STEP.DISKS = 0;
  STEP.OP.REQUESTS = INTEGER(REQUESTS) - 1;
  STEP.DTA_SPACE = 0;
  IF STEP.CORE_SIZE = 60 THEN STEP.CCRE_SIZE = MIN_CORE;
END;
GOTC LOOP;

```

SIM02100
 SIM02110
 SIM02120
 SIM02130
 SIM02140
 SIM02150
 SIM02160
 SIM02170
 SIM02180
 SIM02190
 SIM02200
 SIM02210
 SIM02220
 SIM02230
 SIM02240
 SIM02250
 SIM02260
 SIM02270
 SIM02280
 SIM02290
 SIM02300
 SIM02310
 SIM02320
 SIM02330
 SIM02340
 SIM02350
 SIM02360
 SIM02370
 SIM02380
 SIM02390
 SIM02400
 SIM02410
 SIM02420
 SIM02430
 SIM02440
 SIM02450
 SIM02460
 SIM02470
 SIM02480
 SIM02490
 SIM02500

```

CLASS_B::
DCL  STEPS_B(10) INIT(
DCL  CARDS_B(30) INIT(
DCL  CORE_B(13) INIT(
DCL  TIME_B(40) INIT(
      .3226, .4104, .8871, .9238, .9755,
      .9971, .9996, .9598, .9598, 1.00);
      .4004, .5275, .6002, .6469, .6955,
      .7364, .7781, .8029, .8209, .8516,
      .8651, .8754, .8882, .9006, .9092,
      .9180, .9235, .9321, .9418, .9489,
      .9546, .9578, .9613, .9664, .9704,
      .9750, .9809, .9874, .9916, 1.00);
      .2049, .2056, .2258, .2327, .7194,
      .7403, .7639, .7718, .7725, .9153,
      .9239, .9377, 1.00);
      .0962, .2266, .3625, .4835, .5798,
      .6572, .7121, .7566, .7937, .8235,
      .8478, .8659, .8823, .8966, .9090,
      .9182, .9272, .9340, .9398, .9459,
      .9512, .9555, .9609, .9649, .9681,
      .9715, .9751, .9782, .9807, .9835,
      .9853, .9874, .9892, .9913, .9927,
      .9945, .9959, .9973, .9986, 1.00);

JOB STEPS = INTEGER(STEPS_B);
JOB CARDS = REAL(CARDS_B)*40 + .5;
DO J=1 TO JOB.STEPS;
  STEP_PTR = ADDR(INPUT(INDEX));
  INDEX = INDEX + 1;
  STEP.RUN_TIME = REAL(TIME_B) * 20 + .5;
  STEP.NUMBER = J;
  STEP.CORE_SIZE = INTEGER(CORE_B) * 10 + 50;
  NUM = INTEGER(DEVICES_1);
  STEP.TAPES = TAPE(NUM);
  STEP.DISKS = DISK(NUM);
  STEP.OP_REQUESTS = INTEGER(REQUESTS) - 1;
  STEP.D_A_SPACE = 0;
  IF STEP.CORE_SIZE = 60 THEN STEP.CORE_SIZE = MIN_CORE;
END;
GOTC LOGP;

```

SIM02520
 SIM02530
 SIM02540
 SIM02550
 SIM02560
 SIM02570
 SIM02580
 SIM02590
 SIM02600
 SIM02610
 SIM02620
 SIM02630
 SIM02640
 SIM02650
 SIM02660
 SIM02670
 SIM02680
 SIM02690
 SIM02700
 SIM02710
 SIM02720
 SIM02730
 SIM02740
 SIM02750
 SIM02760
 SIM02770
 SIM02780
 SIM02790
 SIM02800
 SIM02810
 SIM02820
 SIM02830
 SIM02840
 SIM02850
 SIM02860
 SIM02870
 SIM02880
 SIM02890
 SIMC2900
 SIM02910


```

CLASS_C:;
DCL  STEPS_C(11) INIT(
      .5420, .6035, .8912, .9197, .9816,
      .9988, .9994, .9994, .9994, .9997,
      1.00);
DCL  CARDS_C(30) INIT(
      .4205, .5851, .6633, .7262, .7672,
      .7904, .8066, .8447, .8556, .8665,
      .8785, .9036, .9113, .9199, .9272,
      .9341, .9424, .9463, .9477, .9500,
      .9533, .9576, .9632, .9662, .9695,
      .9772, .9901, .9974, .9974, 1.00);
DCL  CORE_C(11) INIT(
      .1516, .1563, .4845, .5063, .5104,
      .6434, .6767, .7594, .8150, .8481,
      1.00);
DCL  TIME_C(38) INIT(
      .1922, .4356, .5415, .6340, .6919,
      .7373, .7730, .8019, .8285, .8475,
      .8681, .8826, .8976, .9058, .9196,
      .9274, .9343, .9413, .9471, .9528,
      .9578, .9625, .9676, .9734, .9771,
      .9800, .9826, .9849, .9870, .9879,
      .9893, .9905, .9922, .9943, .9955,
      .9968, .9985, 1.00);

JOB STEPS = INTEGER(STEPS_C);
JOB CARDS = REAL(CARDS_C) * 40 + .5;
DO J=1 TO JOB STEPS;
  STEP PTR = ADDR(INPUT(INDEX));
  INDEX = INDEX + 1;
  STEP RUN TIME = REAL(TIME_C) * 40 + .5;
  STEP NUMBER = J;
  STEP CORE_SIZE = J;
  NUM = INTEGER(CORE_C) * 20 + 40;
  STEP TAPES = INTEGER(DEVICES_1);
  STEP DISKS = INTEGER(NUM);
  STEP OP REQUESTS = DISK(NUM);
  STEP D A SPACE = TAPE(NUM);
  IF STEP CORE_SIZE = 0 THEN STEP CORE_SIZE = MIN CORE;
  IF STEP CORE_SIZE = 260 THEN STEP CORE_SIZE = 250;
END;
GOTO LOCP;

```

SIM02930
SIM02940
SIM02950
SIM02960
SIM02970
SIM02980
SIM02990
SIM03000
SIM03010
SIM03020
SIM03030
SIM03040
SIM03050
SIM03060
SIM03070
SIM03080
SIM03090
SIM03100
SIM03110
SIM03120
SIM03130
SIM03140
SIM03150
SIM03160
SIM03170
SIM03180
SIM03190
SIM03200
SIM03210
SIM03220
SIM03230
SIM03240
SIM03250
SIM03260
SIM03270
SIM03280
SIM03290
SIM03300
SIM03310
SIM03320
SIM03330
SIM03340

```

CLASS_E:; DCL STEPS_E( 9) INIT(
           .3655, .5241, .5724, .5827, .9930,
           .9965, .9965, .9965, 1.00);
DCL CARDS_E(30) INIT(
           .2769, .5385, .6231, .6538, .7115,
           .7962, .8385, .8846, .9077, .9192,
           .9192, .9385, .9385, .9500, .9538,
           .9615, .9615, .9615, .9654, .5731,
           .9846, .9923, .9923, .9962, .9962,
           .9962, .9962, .9962, 1.00);
DCL CORE_E( 9) INIT(
           .1238, .4492, .4524, .4857, .5174,
           .5730, .8032, .8302, 1.00);
DCL TIME_E(28) INIT(
           .2107, .4700, .6013, .6791, .7391,
           .7812, .8088, .8363, .8541, .8736,
           .8865, .8995, .9125, .9254, .9335,
           .9400, .9481, .9530, .9611, .9643,
           .9724, .9757, .9821, .9870, .9802,
           .9951, .9983, 1.00);
JOB STEPS = INTEGER(STEPS_E);
JOB CARDS = REAL(CARDS_E)*.80 + .5;
DO J=1 TO JOB STEPS;
  STEP_PTR = ADDR(INPUT(INDEX));
  INDEX = INDEX + 1;
  STEP_RUN_TIME = REAL(TIME_E) * 40 + .5;
  STEP_NUMBER = J;
  STEP_CORE_SIZE = INTEGER(CORE_E) * 40 + 20;
  NUM = INTEGER(DEVICES_1);
  STEP_TAPES = TAPE(NUM);
  STEP_DISKS = DISK(NUM);
  STEP_OP_REQUESTS = INTEGER(REQUESTS) - 1;
  STEP_OP_SPACE = 0;
  IF STEP_CORE_SIZE = 60 THEN STEP_CORE_SIZE = MIN_CORE;
  IF STEP_CORE_SIZE = 380 THEN STEP_CORE_SIZE = 350;
END;
GOTO LOOP;

```

IM033360
 SIM033370
 SIM033380
 SIM033390
 SIM033400
 SIM033410
 SIM033420
 SIM033430
 SIM033440
 SIM033450
 SIM033460
 SIM033470
 SIM033480
 SIM033490
 SIM033500
 SIM033510
 SIM033520
 SIM033530
 SIM033540
 SIM033550
 SIM033560
 SIM033570
 SIM033580
 SIM033590
 SIM033600
 SIM033610
 SIM033620
 SIM033630
 SIM033640
 SIM033650
 SIM033660
 SIM033670
 SIM033680
 SIM033690
 SIM033700
 SIM033710
 SIM033720
 SIM033730

```

CLASS_F:;
DCL  STEPS_F( 9) INIT(
      .1352, .1921, .8042, .8149, .9679,
      .9928, .9928, .9964, 1.00);
DCL  CARDS_F(30) INIT(
      .1558, .3116, .3841, .4457, .4819,
      .5399, .5616, .5833, .6123, .6522,
      .7681, .7899, .8080, .8370, .8659,
      .8986, .9058, .9457, .9457, .9674,
      1.00);
DCL  CORE_F(10) INIT(
      .1688, .5775, .6096, .8128, .8277,
      .8977, .9516, .9516, .9745, 1.00);
DCL  TIME_F(30) INIT(
      .3698, .5815, .6813, .7238, .7640,
      .7920, .8090, .8285, .8382, .8540,
      .8662, .8808, .8954, .9100, .9185,
      .9307, .9428, .9538, .9599, .9635,
      .9696, .9732, .9793, .9842, .9851,
      .9903, .9903, .9939, .9976, 1.00);

JOB STEPS = INTEGER(STEPS_F);
JOB CARDS = REAL(CARDS_F) * 80 + .5;
DO J=1 TO JOB.STEPS;
  STEP_PTR = ADDR(INPUT(INDEX));
  INDEX = INDEX + 1;
  STEP.RUN_TIME = REAL(TIME_F) * 40 + .5;
  STEP.NUMBER = J;
  STEP.CORE_SIZE = INTEGER(CORE_F) * 40 + 20;
  NUM = INTEGER(DEVICES_1);
  STEP.TAPES = 0;
  STEP.DISKS = DISK(NUM);
  STEP.OP_REQUESTS = INTEGER(REQUESTS) - 1;
  STEP.DTA_SPACE = 0;
  IF STEP.CORE_SIZE = 63 THEN STEP.CORE_SIZE = MIN_CORE;
  IF STEP.CORE_SIZE = 420 THEN STEP.CORE_SIZE = 400;
END;
GCTC LOOP;

```

SIM03750
 SIM03760
 SIM03770
 SIM03780
 SIM03790
 SIM03800
 SIM03810
 SIM03820
 SIM03830
 SIM03840
 SIM03850
 SIM03860
 SIM03870
 SIM03880
 SIM03890
 SIM03900
 SIM03910
 SIM03920
 SIM03930
 SIM03940
 SIM03950
 SIM03960
 SIM03970
 SIM03980
 SIM03990
 SIM04000
 SIM04010
 SIM04020
 SIM04030
 SIM04040
 SIM04050
 SIM04060
 SIM04070
 SIM04080
 SIM04090
 SIM04100
 SIM04110

```

CLASS_K::
DCL  STEPS_K(15) INIT(
    .3934, .4779, .8934, .5284, .5755,
    .9907, .9953, .9961, .9976, .9976,
    .9584, .9992, .9992, .9992, 1.00);
DCL  CARDS_K(30) INIT(
    .5362, .6292, .7146, .7577, .8000,
    .8377, .8546, .8646, .8715, .8885,
    .9069, .9177, .9262, .9346, .9446,
    .9556, .9577, .9615, .9654, .9731,
    .9800, .9831, .9877, .9923, .9946,
    .9954, .9969, .9977, 1.00);
DCL  CORE_K(20) INIT(
    .1851, .5829, .6330, .7929, .8394,
    .9027, .9282, .9311, .9466, .9615,
    .9696, .9728, .9728, .9503, .9964,
    1.00);
DCL  TIME_K(30) INIT(
    .4677, .6351, .7067, .7579, .7936,
    .8165, .8316, .8657, .8808, .8946,
    .9067, .9145, .9232, .9323, .9407,
    .9465, .9529, .9582, .9623, .9677,
    .9710, .9741, .9781, .9828, .9859,
    .9882, .9909, .9946, .9966, 1.00);

JOB.CLASS = 11;
JOB.STEPS = INTEGER(STEPS_K);
JOB.CARDS = REAL(CARDS_K)*80+.5;
DO J=1 TO JOB.STEPS;
    STEP.PTR = ADDR(INPUT(INDEX));
    INDEX = INDEX + 1;
    STEP.RUN_TIME = REAL(TIME_K)*40+.5;
    STEP.NUMBER = J;
    STEP.CORE_SIZE = INTEGER(CORE_K)*40+20;
    NUM.TAPES = INTEGER(DEVICES_2);
    STEP.DISKS = INTEGER(NUM);
    STEP.OP_REQUESTS = TAPE(NUM);
    STEP.D_A_SPACE = DISK(NUM) - 1;
    IF STEP.CORE_SIZE = 0 THEN STEP.CORE_SIZE = MIN_CORE;
END;
GOTO LOOP;

```

SIM04130
SIM04140
SIM04150
SIM04160
SIM04170
SIM04180
SIM04190
SIM04200
SIM04210
SIM04220
SIM04230
SIM04240
SIM04250
SIM04260
SIM04270
SIM04280
SIM04290
SIM04300
SIM04310
SIM04320
SIM04330
SIM04340
SIM04350
SIM04360
SIM04370
SIM04380
SIM04390
SIM04400
SIM04410
SIM04420
SIM04430
SIM04440
SIM04450
SIM04460
SIM04470
SIM04480
SIM04490
SIM04500
SIM04510
SIM04520
SIM04530


```

CLASS_C::
DCL  STEPS_0( 2) INIT( .5524, 1.00);
JOB. STEPS = INTEGER(STEPS_0);
JOB. CARDS = REAL(CARDS_A) * 40 + .5;
DO J=1 TO JOB. STEPS;
  STEP. PTR = ADDR(INPUT(INDEX));
  INDEX = INDEX + 1;
  STEP. RUN TIME = REAL(TIME_A) * 10 + .5;
  STEP. NUMBER = J;
  STEP. CORE SIZE = INTEGER(CORE_A) * 10 + 50;
  STEP. TAPES = 0;
  STEP. DISKS = 0;
  STEP. OP REQUESTS = 0;
  STEP. DATA SPACE = 0;
  IF STEP. CORE_SIZE = 60 THEN STEP. CCRE_SIZE = MIN_CORE;
END;
GOTO LOOP;

LOOP:  END;
      END GENERATE_JOBS;

(NOFIXEDOVERFLOW):
RANDU: PROCEDURE(I1,I2,X);
/* RANDOM NUMBER GENERATOR */
DCL  I1 FIXED BIN(31);
      I2 FIXED BIN(31);
      X;
I2 = I1 * 65539;
IF I2 < 0 THEN I2 = I2 + 2147483647 + 1;
X = I2;
X = X * .4656613E-9;
END RANDU;

```

SIM04550
SIM04560
SIM04570
SIM04580
SIM04590
SIM04600
SIM04610
SIM04620
SIM04630
SIM04640
SIM04650
SIM04660
SIM04670
SIM04680
SIM04690
SIM04700
SIM04710
SIM04720
SIM04730
SIM04740
SIM04750
SIM04760
SIM04770
SIM04780
SIM04790
SIM04800
SIM04809
SIM04810
SIM04820
SIM04830
SIM04840
SIM04850
SIM04860
SIM04870
SIM04880
SIM04890
SIM04900
SIM04910
SIM04920
SIM04930

```

GET_MAIN:  PROCEDURE(1, CORESIZE);
/* ALLOCATE MAIN STORAGE */
DCL  CORESIZE  FIXED BIN,
     N         FIXED BIN,
     I         FIXED BIN;

CORE_HIGH(1) = 0;
CORE_LOW(1)  = 0;
N = 1;
DO WHILE (FREE_CORE(N) - CORE(N) /= 0);
  IF FREE_CORE(N) - CORE(N) >= CORESIZE
  THEN
    DO;
      CORE_HIGH(1) = FREE_CORE(N);
      FREE_CORE(N) = FREE_CORE(N) - CORESIZE;
      CORE_LOW(1)  = FREE_CORE(N);
      HIGH(17)     = CORE_HIGH(1);
      LOW(17)      = CORE_LOW(1);
      INIT(17)     = 1;
      N = 1;
    DO WHILE (HIGH(N) /= 0);
      IF HIGH(N) < HIGH(17)
      THEN
        DO;
          USED_CORE(18) = USED_CORE(N);
          USED_CORE(N)  = USED_CORE(17);
          USED_CORE(17) = USED_CORE(18);
        END;
        N = N + 1;
      END;
      USED_CORE(N) = USED_CORE(17);
      N = N + 2;
    END;
  END GET_MAIN;

```

```

SIM04950
SIM04960
SIM04970
SIM04980
SIM04990
SIM05000
SIM05010
SIM05020
SIM05030
SIM05040
SIM05050
SIM05060
SIM05070
SIM05080
SIM05090
SIM05100
SIM05110
SIM05120
SIM05130
SIM05140
SIM05150
SIM05160
SIM05170
SIM05180
SIM05190
SIM05200
SIM05210
SIM05220
SIM05230
SIM05240
SIM05250
SIM05260
SIM05270
SIM05280
SIM05290
SIM05300
SIM05310
SIM05320

```

```

FREE_MAIN:  PROCEDURE(I);
/* FREE MAIN STORAGE */

DCL  COMP FIXED BIN,
     SAVE FIXED BIN,
     N FIXED BIN,
     I FIXED BIN;

DO COMP = CORE_HIGH(I), CORE_LCW(I);
  N = 1;
  DO WHILE (FREE_CORE(N) >= 0);
    IF FREE_CORE(N) = COMP
      THEN FREE_CORE(N), COMP = FREE_CORE(N+1);
    IF FREE_CORE(N) < COMP
      THEN
        DO;
          SAVE = FREE_CORE(N);
          FREE_CORE(N) = COMP;
          COMP = SAVE;
        END;
        N = N + 1;
      END;
    FREE_CORE(N) = COMP;
  END;
  COMP = CORE_HIGH(I);
  N = 1;
  DO WHILE (HIGH(N) >= 0);
    IF HIGH(N) = COMP
      THEN
        DO;
          USED_CORE(N) = USED_CORE(N+1);
          COMP = HIGH(N+1);
        END;
        N = N + 1;
      END;
  END;
END;

END FREE_MAIN;

```

```

SIM05340
SIM05350
SIM05360
SIM05370
SIM05380
SIM05390
SIM05400
SIM05410
SIM05420
SIM05430
SIM05440
SIM05450
SIM05460
SIM05470
SIM05480
SIM05490
SIM05500
SIM05510
SIM05520
SIM05530
SIM05540
SIM05550
SIM05560
SIM05570
SIM05580
SIM05590
SIM05600
SIM05610
SIM05620
SIM05630
SIM05640
SIM05650
SIM05660
SIM05670
SIM05680
SIM05690
SIM05700

```

```
INIT_QUEUE: PROCEDURE;
/* INITIALIZE THE JOB INPUT QUEUE */
```

```
  DCL I FIXED BIN;
  DO I=1 TO JOB_QUEUE_SIZE;
    JOB_LINK(I) = I + 1;
  END;
  LINK(JOB_QUEUE_SIZE) = 0;
  FREE_HEAD = 1;
  AMOUNT = 0;
  HEAD = 0;
  END INIT_QUEUE;
```

```
ALLCC: PROCEDURE;
/* ALLOCATE SPACE IN THE JOB INPUT QUEUE */
```

```
  DCL I FIXED BIN;
  I = FREE_HEAD;
  IF I = 0 THEN /* SIMULATION QUEUE FULL */
    DO;
      PUT SKIP LIST ('SIMULATION QUEUE FULL. RESTART. ');
      PUT SKIP;
      GOTO SUPERVISOR;
    END;
  FREE_HEAD = JOB_LINK(I);
  RETURN(I);
  END ALLCC;
```

```
SIM05720
SIM05730
SIM05740
SIM05750
SIM05760
SIM05770
SIM05780
SIM05790
SIM05800
SIM05810
SIM05820
SIM05830
SIM05840
SIM05850
SIM05860
SIM05870
SIM05880
SIM05890
SIM05900
SIM05910
SIM05920
SIM05930
SIM05940
SIM05950
SIM05960
SIM05970
SIM05980
SIM05990
SIM06000
SIM06010
SIM06020
SIM06030
SIM06040
SIM06050
```



```

SIM06070
SIM06080
SIM06090
SIM06100
SIM06110
SIM06120
SIM06130
SIM06140
SIM06150
SIM06160
SIM06170
SIM06180
SIM06190
SIM06200
SIM06210
SIM06220
SIM06230
SIM06240
SIM06250
SIM06260
SIM06270
SIM06280
SIM06290
SIM06300
SIM06310
SIM06320
SIM06330
SIM06340
SIM06350
SIM06360
SIM06370
SIM06380
SIM06390
SIM06400
SIM06410
SIM06420
SIM06430
SIM06440
SIM06450
SIM06460
SIM06470
SIM06480

```

```

ENCQUEUE: PROCEDURE;
/* ENQUEUE A NEW JOB INTO THE JOB QUEUE */
/* ACCORDING TO IT'S CLASS AND PRIORITY */

DCL NEXT-IN FIXED BIN,
NEXT-OUT FIXED BIN,
LAST IN FIXED BIN,
NEW NEW;

/* FIND POSITION IN QUEUE */
AMOUNT(JOB.CLASS) = AMOUNT(JOB.CLASS) + 1;
IN = JOBS(JOB-COUNT);
NEXT = HEAD(JOB.CLASS);
LAST = 0;
DO WHILE (NEXT .NE. 0);
IF JOB.PRIORITY <= QUEUE.PRIORITY(NEXT)
THEN
DO;
LAST = NEXT;
NEXT = JOB_LINK(NEXT);
END;
ELSE NEXT = 0;
END;

/*ENQUEUE JOB DESCRIPTION INTO GIVEN POSITION */
NEW = ALLOC;
IF LAST = 0
THEN
DO;
JOB_LINK(NEW) = HEAD(JOB.CLASS);
HEAD(JOB.CLASS) = NEW;
END;
ELSE /* JOB GOES ELSEWHERE IN THE QUEUE */
DO;
JOB_LINK(NEW) = JOB_LINK(LAST);
JOB_LINK(LAST) = NEW;
END;
QUEUE(NEW) = INPUT(IN);
QUEUE.ARRIVAL(NEW) = TIME.SYSTEM;

```

```

/* ENQUEUE ALL STEP DESCRIPTIONS */
DO I=1 TO JOB.STEPS;
  STEP_LINK(NEW) = ALLOC;
  NEW = STEP_LINK(NEW);
  QUEUE(NEW) = INPUT(IN + I);
END;
STEP_LINK(NEW) = 0;
END ENQUEUE;

```

```

DEQUEUE: PROCEDURE(I);
/* DEQUEUE A JOB */

```

```

DCL N FIXED BIN;
M FIXED BIN;
I FIXED BIN;

```

```

JOB_INDEX(I) = 0;

```

```

DO N=1 TO 8;
  M = IN(I).CLASS(N);
  IF M ^= 0
    THEN
      DO;
        IF HEAD(M) ^= 0 /* JOB FOUND */
          THEN
            DO;
              JOB_INDEX(I) = HEAD(M);
              STEP_LINK(I) = STEP_LINK(HEAD(M));
              JOB_NO(I) = QUEUE.NUMBER(JOB_INDEX(I));
              HEAD(M) = JOB_LINK(HEAD(M));
              AMOUNT(M) = AMOUNT(M) - 1;
              N
            END;
          END;
        END;
      END DEQUEUE;

```

```

SIM06500
SIM06510
SIM06520
SIM06530
SIM06540
SIM06550
SIM06560
SIM06570
SIM06580
SIM06590
SIM06600
SIM06610
SIM06620
SIM06630
SIM06640
SIM06650
SIM06660
SIM06670
SIM06680
SIM06690
SIM06700
SIM06710
SIM06720
SIM06730
SIM06740
SIM06750
SIM06760
SIM06770
SIM06780
SIM06790
SIM06800
SIM06810
SIM06820
SIM06830
SIM06840
SIM06850
SIM06860
SIM06870
SIM06880
SIM06890

```

```

FREE_QUEUE: PROCEDURE(I);
/* ADD QUEUE SPACE TO THE FREE LIST */
JOB_LINK(JOB_INDEX(I)) = FREE_HEAD;
FREE_HEAD = JOB_INDEX(I);
DO WHILE (STEP_LINK(FREE_HEAD) = 0);
  JOB_LINK(STEP_LINK(FREE_HEAD)) = FREE_HEAD;
  FREE_HEAD = STEP_LINK(FREE_HEAD);
END;

END FREE_QUEUE;

(NOFIXEDOVERFLOW):
CHANGE_SEED: PROCEDURE;
/* CHANGE SEED OF RANDOM NUMBER GENERATOR */
DCL 1 CLOCK, CHAR(8),
2 DATE, CHAR(8),
2 TIME, FIXED BIN(31),
2 V_TIME, FIXED BIN(31);

CALL IXECCLK(CLOCK);
V_TIME = V_TIME * 10;
SEED_1 = V_TIME + SEED_1;
SEED_2 = V_TIME + SEED_2;

END CHANGE_SEED;

```

```

SIM06910
SIM06920
SIM06930
SIM06940
SIM06950
SIM06960
SIM06970
SIM06980
SIM06990
SIM07000
SIM07010
SIM07416
SIM07417
SIM07418
SIM07419
SIM07420
SIM07430
SIM07440
SIM07450
SIM07460
SIM07470
SIM07480
SIM07490
SIM07500
SIM07510
SIM07520
SIM07530
SIM07540
SIM07550
SIM07560

```

```

START: PROCEDURE;
/* INITIALIZE FOR A NEW RUN */

DCL 1 MOD,
    2 SPOOL,
    2 SPACE_H,
    2 CORE_H,
    2 CORE_L,
    2 DISKS,
    2 TAPES;

DCL N
    FIXED BIN;

TIME LC = 0;
IN_QUEUE = 0;
N = 0;
SYSTEM_FREE_CORE = 0;
PUT SKIP LIST ('SYSTEM MODIFICATIONS? (1=YES, 0=NO)');
GET LIST (N);
IF N=1 THEN
DO;
MOD = INITIAL_VALUES;
PUT SKIP LIST ('ENTER:');
GET DATA;
INITIAL_VALUES = MOD;
END;
SYSTEM_TAPES = I_TAPES;
SYSTEM_D_A_SPACE = I_D_A_SPACE;
SYSTEM_USED_CORE = 0;
SYSTEM_FREE_CORE(1) = I_CORE_HIGH;
SYSTEM_FREE_CORE(2) = I_CORE_LOW;
SYSTEM_DISKS = I_DISKS;
SYSTEM_IN_SPOOL = I_IN_SPOOL;
STATUS = 2;
RESOURCES = 0;
DO I=1 TO 15;
IF STOP_SWITCH(I) >= 0 THEN STOP_SWITCH(I) = 0;
ELSE STOP_SWITCH(I) = -2;
END;
CALL INIT_QUEUE;
END START;

```

```

SIM07030
SIM07040
SIM07050
SIM07051
SIM07052
SIM07053
SIM07054
SIM07055
SIM07056
SIM07057
SIM07058
SIM07060
SIM07070
SIM07080
SIM07090
SIM07100
SIM07110
SIM07120
SIM07130
SIM07140
SIM07150
SIM07160
SIM07170
SIM07180
SIM07190
SIM07200
SIM07210
SIM07220
SIM07230
SIM07240
SIM07250
SIM07260
SIM07270
SIM07280
SIM07290
SIM07300
SIM07310
SIM07320
SIM07330
SIM07340
SIM07350
SIM07360
SIM07370
SIM07380
SIM07390
SIM07400

```



```

SET_SIM_PARAMETERS: PROCEDURE;
/* ALLOW RUN PARAMETERS TO BE ENTERED */
PUT SKIP LIST ('RUN PARAMETERS:');
PUT SKIP LIST ('TIME (0 < SEC. < 604801) :');
PUT SKIP;
GET LIST (DELTA_TIME);
DO WHILE (DELTA_TIME < 1 | DELTA_TIME > 604800);
  PUT LIST (DELTA_TIME);
  PUT SKIP;
  GET LIST (DELTA_TIME);
END;
PUT SKIP LIST ('JOBS (0 < NUMBER < 1001) :');
PUT SKIP;
GET LIST (MAX_JOBS);
DO WHILE (MAX_JOBS < 1 | MAX_JOBS > 1000);
  PUT LIST (MAX_JOBS);
  PUT SKIP;
  GET LIST (MAX_JOBS);
END;
END SET_SIM_PARAMETERS;

```

```

SET_RESTART: PROCEDURE;
PUT SKIP;
PUT LIST ('RESTART PARAM. (0 - 6) : '); SKIP;
PUT SKIP;
GET LIST (RESTART);
DO WHILE (RESTART < 0 | RESTART > 6);
  PUT LIST ('INVALID INPUT. RETRY: '); SKIP;
  PUT SKIP;
  GET LIST (RESTART);
END;
END SET_RESTART;

```

SIM07580
SIM07590
SIM07600
SIM07610
SIM07620
SIM07630
SIM07640
SIM07650
SIM07660
SIM07670
SIM07680
SIM07690
SIM07700
SIM07710
SIM07720
SIM07730
SIM07740
SIM07750
SIM07760
SIM07770
SIM07780
SIM07790
SIM07800

SIM07810
SIM07820
SIM07830
SIM07840
SIM07850
SIM07860
SIM07870
SIM07880
SIM07890
SIM07900
SIM07910
SIM07920
SIM07930
SIM07940
SIM07950
SIM07960

```

SIM07980
SIM07990
SIM08000
SIM08010
SIM08020
SIM08030
SIM08040
SIM08050
SIM08060
SIM08070
SIM08080
SIM08090
SIM08100
SIM08110
SIM08120
SIM08130
SIM08140
SIM08150
SIM08160
SIM08170
SIM08180
SIM08190
SIM08200
SIM08210
SIM08220
SIM08230
SIM08240
SIM08250
SIM08260
SIM08270
SIM08280
SIM08290
SIM08300
SIM08310
SIM08320

```

```

INITIALIZE: PROCEDURE;
/* INITIALIZE THE SIMULATION MODEL */

INITIALIZE_1: ENTRY;

OPEN FILE (PI) PAGESIZE(75);
SEED_1 = 98765;
SEED_2 = 12345;
ALPHA = 1.3083;
STATISTICS.MAXI = 0;
STATISTICS.NO = 0;
STOP SWITCH = -2;
IN_CLASSES = 0;
CLASSES = 0;
RETURN;

INITIALIZE_2: ENTRY;

IF RESTART = 3 | RESTART = 6 /* CHANGE SEED FOR JOB GEN. */
THEN CALL CHANGE_SEED;
IF RESTART < 4 /* EMPTY QUEUE FOR NEW RUN */
THEN CALL START;
IF RESTART /= 2 & RESTART /= 5 /* GET SIM. PARAMETERS */
THEN /* GENERATE NEW JOB STREAM */
DO;
CALL SET SIM PARAMETERS;
CALL GENERATE_JOBS;
END;
TIME.READER = TIME.SYSTEM + INPUT.DEL_ARRIVAL(1);
JOB_CCUNT = 1;
JOB_PTR = ADDR(INPUT);
TIME.STOP = TIME.SYSTEM + DELTA_TIME;
RETURN;

END INITIALIZE;

```

```

MODIFY:  PROCEDURE;
/* ALLOW INITIATOR MODIFICATIONS TO BE ENTERED */
DCL N CHECK
      TEMP
      I
      FIXED BIN;
      FIXED BIN;
      FIXED BIN;
      FIXED BIN;

LCCP:
      PUT SKIP LIST ('MODIFY INITIATORS:');
      PUT SKIP LIST ('ENTER N (1-15 = INIT.# , 0 = END MOD.) :');
      GET LIST (N);
      IF N < 1 | N > 15 THEN GOTO SET_TRACE;
      PUT SKIP LIST ('ENTER JOB CLASSES 'A-O' OR 'STOP' :');
      PUT SKIP LIST (CLASSES(N));
      GET LIST (CLASSES(N));
      CHECK = VERIFY(CLASSES(N), 'ABCDEFGHIJKLMNO ');
      IF CHECK = 0 THEN /* STOP INITIATOR */
      DO;
      CLASSES(N) = ' STOP ';
      IF STOP_SWITCH(N) >= 0
      THEN STOP_SWITCH(N) = STOP_SWITCH(N) - 2;
      END;
      ELSE /* SET NEW JOB CLASSES */
      DO;
      IF STOP_SWITCH(N) = -2
      THEN TIME LC(N) = TIME.SYSTEM;
      IF STOP_SWITCH(N) = 0 | STOP_SWITCH(N) = -2
      THEN TIME.INITIATOR(N) = TIME.SYSTEM;
      IF STOP_SWITCH(N) < 0
      THEN STOP_SWITCH(N) = STOP_SWITCH(N) + 2;
      DO I=1 TO 8;
      TEMP = UNSPEC(SUBSTR(CLASSES(N), I, 1));
      IF TEMP < 202
      THEN TEMP = TEMP - 192;
      ELSE TEMP = TEMP - 199;
      IF TEMP < 0
      THEN TEMP = 0;
      IN(N).CLASS(I) = TEMP;
      END;
      END;
      GOTC LOOP;

```

```

SIM08340
SSIM08350
SSIM08360
SSIM08370
SSIM08380
SSIM08390
SSIM08400
SSIM08410
SSIM08420
SSIM08430
SSIM08440
SSIM08450
SSIM08460
SSIM08470
SSIM08480
SSIM08490
SSIM08500
SSIM08510
SSIM08520
SSIM08530
SSIM08540
SSIM08550
SSIM08560
SSIM08570
SSIM08580
SSIM08590
SSIM08600
SSIM08610
SSIM08620
SSIM08630
SSIM08640
SSIM08650
SSIM08660
SSIM08670
SSIM08680
SSIM08690
SSIM08700
SSIM08710
SSIM08720
SSIM08730
SSIM08740
SSIM08750
SSIM08760
SSIM08770
SSIM08780

```

```

SET_TRACE:
PUT SKIP LIST ('SET TRACE, CORE MAP, STAT. PARAM. (1=CN) :');
PUT SKIP;
GET LIST (TRACE, CORE_MAP, RESULTS);
IF TRACE = 1
THEN PUT FILE (P1) PAGE LINE(9) EDIT ('* ') (X(15),A(3));
END MODIFY;

CF_RESPONSE: PROCEDURE(I);
/* SIMULATE OPERATOR RESPONSE */
DCL DISK_MOUNT(31) INIT(
.0000, .0197, .0724, .1513, .2829,
.3487, .4868, .5724, .6250, .6842,
.7434, .7566, .7961, .8092, .8355,
.8487, .8618, .8816, .9079, .9276,
.9342, .9403, .9474, .9539, .9605,
.9671, .9737, .9803, .9868, .9934,
1.00);
DCL TAPE_MOUNT(31) INIT(
.0000, .0733, .2454, .4249, .5421,
.6264, .6996, .7253, .7509, .7839,
.8168, .8352, .8681, .8628, .8938,
.9121, .9267, .9377, .9451, .9524,
.9457, .9670, .9707, .9744, .9780,
.9817, .9853, .9890, .9927, .9963,
1.00);
DCL ANSWER(11) INIT(
.6715, .6968, .7401, .7578, .8159,
.8267, .8484, .8626, .8700, .8917,
1.00);
DCL CANCEL(2) INIT( .0130, 1.00);
DCL X;

```

```

SIM08800
SIM08810
SIM08820
SIM08830
SIM08840
SIM08850
SIM08860
SIM08870
SIM08880
SIM08890
SIM08900
SIM08910
SIM08920
SIM08930
SIM08940
SIM08950
SIM08960
SIM08970
SIM08980
SIM08990
SIM09000
SIM09010
SIM09020
SIM09030
SIM09040
SIM09050
SIM09060
SIM09070
SIM09080
SIM09090
SIM09100
SIM09110
SIM09120
SIM09130
SIM09140
SIM09150
SIM09160

```


REAL2: PROCEDURE(TABLE);

DCL TABLE(60), NUMBER, VALUE, RANDOM_X, X_LOW;

CALL RANDU(SEED_2, SEED_2, RANDCM_X);

NUMBER = 1;

DO WHILE (RANDCM_X > TABLE(NUMBER));

NUMBER = NUMBER + 1;

END;

IF NUMBER = 1 THEN X_LOW = 0;

ELSE X_LOW = TABLE(NUMBER-1);

VALUE = NUMBER - 1 + (RANDOM_X - X_LOW)

/ (TABLE(NUMBER) - X_LOW);

RETURN(VALUE);

END REAL2;

OP_TIME(I) = 0;

OP_ANSWER(I) = REAL2(CANCEL);

IF STEP.DISKS > I.DISKS | STEP.TAPES > I.TAPES

THEN OP_ANSWER(I) = 0;

IF OP_ANSWER(I) = 0 THEN OP_ANSWER(I) = -1;

IF STEP.OP_REQUESTS /= 0 | CP_ANSWER(I) < 0

THEN

DO;

X = REAL2(ANSWER);

IF X < 10 THEN OP_TIME(I) = X * 20;

ELSE OP_TIME(I) = (X - 10) * 620 + 200;

END;

IF OP_ANSWER(I) < 0 | STATUS(I) = 4 THEN RETURN;

IF STEP.TAPES /= 0 THEN OP_TIME(I) = REAL2(TAPE_MOUNT)

* 20 * STEP.TAPES + .5 * OP_TIME(I);

IF STEP.DISKS /= 0 THEN OP_TIME(I) = REAL2(DISK_MOUNT)

* 20 * STEP.DISKS + .5 * OP_TIME(I);

IF OP_TIME(I) > 1000 THEN OP_TIME(I) = 1000;

END OP_RESPONSE;

SIM09180
SIM09190
SIM09200
SIM09210
SIM09220
SIM09230
SIM09240
SIM09250
SIM09260
SIM09270
SIM09280
SIM09290
SIM09300
SIM09310
SIM09320
SIM09330
SIM09340
SIM09350
SIM09360
SIM09370
SIM09380
SIM09382
SIM09384
SIM09390
SIM09400
SIM09410
SIM09420
SIM09430
SIM09440
SIM09450
SIM09460
SIM09470
SIM09480
SIM09490
SIM09500
SIM09510
SIM09520
SIM09530
SIM09540

```

WRITE_STATISTICS: PROCEDURE;
/* WRITE STATISTICS */
DCL SAVE_1 FIXED BIN,
      SAVE_2 FIXED BIN,
      I N
      N
TIME.SYSTEM = MIN(TIME.SYSTEM, TIME.STOP);
IF RESULTS = 1 /* GATHER STATISTICAL RESULTS */
THEN
DO;
TIME SNAP = TIME.SYSTEM;
SAVE_1 = TRACE;
DO I=1 TC 15;
SAVE_2 = OP ANSWER(I);
IF STOP SWITCH(I) = -2 THEN CALL STATISTIC_END_WW(I);
IF STATUS(I) = 1 THEN CALL STATISTIC_END_WV(I);
IF STATUS(I) = 2 THEN CALL STATISTIC_END_WS(I);
IF STATUS(I) = 4 THEN CALL STATISTIC_END_WM(I);
IF STATUS(I) = 5 THEN CALL STATISTIC_END_WL(I);
IF STATUS(I) = 6 THEN CALL STATISTIC_END_WV(I);
IF STATUS(I) = 7 THEN CALL STATISTIC_END_WS(I);
UP ANSWER(I) = SAVE_2;
TIME LC(I) = TIME.SYSTEM;
END;
WRITE FILE (STAT) FROM (STATISTICS);
STATISTICS.MAXI = 0;
STATISTICS.NO = 0;
IN QUEUE = AMCUNT;
TRACE = SAVE_1;
END;

NEXT_I:

```

```

IM09560
SIM09570
SIM09580
SIM09590
SIM09600
SIM09610
SIM09620
SIM09630
SIM09640
SIM09650
SIM09660
SIM09670
SIM09680
SIM09690
SIM09700
SIM09710
SIM09720
SIM09730
SIM09740
SIM09750
SIM09760
SIM09770
SIM09780
SIM09790
SIM09800
SIM09810
SIM09820
SIM09830
SIM09840
SIM09850
SIM09860
SIM09870
SIM09880

```

```

IF CORE MAP ^= 1 THEN RETURN;
/* PRINT OUT A CORE MAP */

PUT FILE (P1) SKIP(6) EDIT (' USAGE OF MAIN MEMORY')
PUT FILE (P1);
PUT FILE (P1) SKIP(2) EDIT (' HIGH LCM CORE INIT.# JOB #')
PUT FILE (P1);
PUT FILE (P1) SKIP(2);
IF HIGH(1) = 0
THEN
DO;
CORE = I_CORE_HIGH - I_CORE_LOW;
PUT FILE (P1) SKIP EDIT
(I_CORE_HIGH, I_CORE_LOW, CORE, ' FREE')
(X(15), 3 F(5), X(3), A(5));
RETURN;
END;

CORE = I_CORE_HIGH - HIGH(1);
IF HIGH(1) < I_CORE_HIGH THEN PUT FILE (P1) SKIP EDIT
(I_CORE_HIGH, HIGH(1), CORE, 'FREE')(X(15), 3 F(5), X(4), A(4));
N = 1;
DO WHILE (HIGH(N) ^= 0);
CORE = HIGH(N) - LOW(N);
PUT FILE (P1) SKIP EDIT (HIGH(N), LOW(N), CORE, INIT(N),
JOB NO (INIT(N)))(X(15), 3 F(5), X(3), 2 F(5));
IF LOW(N) ^= HIGH(N+1) & HIGH(N+1) ^= 0
THEN
DO;
CORE = LOW(N) - HIGH(N+1);
PUT FILE (P1) SKIP EDIT
(LOW(N), HIGH(N+1), CORE, ' FREE')
(X(15), 3 F(5), X(3), A(5));
END;
N = N + 1;
END;
IF LOW(N-1) > I_CORE_LOW
THEN
DO;
CORE = LOW(N-1) - I_CORE_LOW;
PUT FILE (P1) SKIP EDIT
(LOW(N-1), I_CORE_LOW, CORE, 'FREE')(X(15), 3 F(5), X(4), A(4));
END;
RETURN;
END WRITE_STATISTICS;

```

SIM09900
SIM09910
SIM09920
SIM09930
SIM09940
SIM09950
SIM09960
SIM09970
SIM09980
SIM09990
SIM10000
SIM10010
SIM10020
SIM10030
SIM10040
SIM10050
SIM10060
SIM10070
SIM10080
SIM10090
SIM10100
SIM10110
SIM10120
SIM10130
SIM10140
SIM10150
SIM10160
SIM10170
SIM10180
SIM10190
SIM10200
SIM10210
SIM10220
SIM10230
SIM10240
SIM10250
SIM10260
SIM10270
SIM10280
SIM10290
SIM10300
SIM10310
SIM10320
SIM10330
SIM10340

```

STATISTIC: PRCCEDURE;
/* COLLECT STATISTICAL DATA */
DCL WAIT FIXED BIN;
I

```

SIM10360
SIM10370
SIM10380
SIM10390
SIM10400
SIM10410
SIM10420
SIM10430
SIM10440
SIM10450
SIM10460
SIM10470
SIM10480
SIM10490
SIM10500
SIM10510
SIM10520
SIM10530
SIM10540
SIM10550
SIM10560
SIM10570
SIM10580
SIM10590
SIM10600
SIM10610
SIM10620
SIM10630
SIM10640
SIM10650
SIM10660
SIM10670
SIM10680
SIM10690
SIM10700
SIM10710
SIM10720
SIM10730
SIM10740
SIM10750
SIM10760
SIM10770

```

STATISTIC START JOB: ENTRY(I);
/* JOB STARTED */
JOB START(I, JOB.CLASS) = JOB_START(I, JOB.CLASS) + 1;
WAIT = TIME.SYSTEM - JCB.ARRIVAL;
JOB_WAIT(I, JOB.CLASS) = JOB_WAIT(I, JOB.CLASS) + WAIT;
JOB_WAIT(I, JOB.CLASS) = JOB_WAIT(I, JOB.CLASS) + WAIT * 2;
JOB_WAIT(I, JOB.CLASS) = MAX(JOB_WAIT(I, JOB.CLASS), WAIT);
IF TRACE = 1
THEN PUT FILE (P1) EDIT
( TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' STARTED BY INITIATOR ',
I, ' * ' ) (F(7), A(6), F(6), A(22), F(2), SKIP, X(15), A(3));
RETURN;

STATISTIC END STEP: ENTRY(I);
/* UPDATE TOTAL TIME SERVING JOBS */
WAIT = TIME.SYSTEM - TIME.LC(I);
MAX_SJ(I) = MAX(MAX_SJ(I), WAIT);
TIME_SJ(I) = TIME_SJ(I) + WAIT;
RETURN;

STATISTIC END JOB: ENTRY(I);
/* JOB TERMINATED */
IF TRACE = 1
THEN PUT FILE (P1) EDIT
( TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' TERMINATED', ' * ' )
(F(7), A(6), F(6), A(11), SKIP, X(15), A(3));
RETURN;

```



```

STATISTIC_BEGIN WM: ENTRY(I);
/* NO JOBS TO SERVE. WAITING FOR WORK */

IF TRACE = 1
  THEN PUT FILE (PI) EDIT
  (TIME.SYSTEM, ' * INIT. ', I, ' WAITING FOR WORK', '** ')
  (F(7), A(10), F(2), A(17), SKIP, X(15), A(3));
RETURN;

STATISTIC_END WM: ENTRY(I);
/* UPDATE TOTAL TIME WAITING FOR WORK */

WAIT = TIME.SYSTEM - TIME.LC(I);
MAX_WM(I) = MAX (MAX_WM(I), WAIT);
TIME_WM(I) = TIME_WM(I) + WAIT;
RETURN;

STATISTIC_END WM: ENTRY(I);
/* UPDATE TOTAL TIME WAITING FOR MAIN MEMORY */

WAIT = TIME.SYSTEM - TIME.LC(I);
MAX_WM(I) = MAX (MAX_WM(I), WAIT);
TIME_WM(I) = TIME_WM(I) + WAIT;
RETURN;

```

```

SIM10820
SIM10830
SIM10840
SIM10850
SIM10860
SIM10870
SIM10880
SIM10890
SIM10900
SIM10910
SIM10920
SIM10930
SIM10940
SIM10950
SIM10960
SIM10970
SIM10980
SIM10990
SIM11000
SIM11010
SIM11020
SIM11030
SIM11040
SIM11050
SIM11060
SIM11070
SIM11080
SIM11090

```

SIM111110
SIM111120
SIM111130
SIM111140
SIM111150
SIM111160
SIM111170
SIM111180
SIM111190
SIM111200
SIM111210
SIM111220
SIM111230
SIM111240
SIM111250
SIM111260
SIM111270
SIM111280
SIM111290
SIM111300
SIM111310
SIM111320
SIM111330
SIM111340
SIM111350
SIM111360
SIM111370
SIM111380
SIM111390
SIM111400
SIM111410
SIM111420
SIM111430
SIM111440
SIM111450
SIM111460
SIM111470
SIM111480
SIM111490

```

STATISTIC_BEGIN WD: ENTRY(I);
/* NOT ALL DEVICES AVAILABLE. SIMULATE */
/* OPERATOR RESPONSE TIME AND ANSWER */

CALL OP_RESPONSE(I);
IF TRACE = 1 & STEP.DISKS /= 0
  THEN PUT FILE (P1) EDIT
  (TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' WAITING FOR ',
  STEP.DISKS, ' DISK(S)', '* ');
  (F(7), A(6), F(6), A(13), F(2), A(10), SKIP, X(15), A(3));
IF TRACE = 1 & STEP.TAPES /= 0
  THEN PUT FILE (P1) EDIT
  (TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' WAITING FOR ',
  STEP.TAPES, ' TAPE(S)', '* ');
  (F(7), A(6), F(6), A(13), F(2), A(10), SKIP, X(15), A(3));
RETURN;

STATISTIC_END WD: ENTRY(I);
/* UPDATE TOTAL TIME WAITING FOR DEVICES */

WAIT = TIME.SYSTEM - TIME.LC(I);
MAX_WD(I) = MAX (MAX_WD(I), WAIT);
TIME_WD(I) = TIME_WD(I) + WAIT;
OP_ANSWER(I) = 0;
RETURN;

STATISTIC_ABEND_DEVICE: ENTRY(I);
/* JOB CANCELLED BY OPERATOR BECAUSE OF LACK OF DEVICES */
ABEND_DEVICE(I) = ABEND_DEVICE(I) + 1;
IF TRACE = 1
  THEN PUT FILE (P1) EDIT
  (TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' CANCELLED BY OPERATOR ',
  '* ');
  (F(7), A(6), F(6), A(13), F(2), A(10), SKIP, X(15), A(3));
RETURN;

```

```

STATISTIC BEGIN WV: ENTRY(I); SIMULATE OPERATOR RESPONSE TIME */
/* ALLOCATE DATA SETS.

CALL OP_RESPONSE(I);
IF TRACE = 1 & STEP.TAPES ^= 0
THEN PUT FILE (PI) EDIT
(TIME.SYSTEM, ' * JOB', JCB.NUMBER, ' : MOUNT ',
STEP.TAPES, ' TAPE(S)', '* ');
(F(7), A(6), F(6), A(9), F(2), A(9), SKIP, X(15), A(3));
IF TRACE = 1 & STEP.DISKS ^= 0
THEN PUT FILE (PI) EDIT
(TIME.SYSTEM, ' * JOB', JCB.NUMBER, ' : MOUNT ',
STEP.DISKS, ' DISK(S)', '* ');
(F(7), A(6), F(6), A(9), F(2), A(9), SKIP, X(15), A(3));
IF TRACE = 1 & STEP.OP_REQUESTS ^= 0
THEN PUT FILE (PI) EDIT
(TIME.SYSTEM, ' * JOB', JCB.NUMBER, ' : ALLOCATE 1 ',
'DATA SET', '* ');
(F(7), A(6), F(6), A(14), A(8), SKIP, X(15), A(3));
RETURN;

STATISTIC END WV: ENTRY(I);
/* UPDATE TOTAL TIME WAITING FOR DATA SET ALLOCATION */

OP_ANSWER(I) = 0;
WAIT = TIME.SYSTEM - TIME.LC(I);
MAX_WV(I) = MAX(MAX_WV(I), WAIT);
TIME_WV(I) = TIME.WV(I) + WAIT;
IF TRACE = 1 & (IN.TAPES(I) ^= 0 | IN.DISKS(I) ^= 0)
THEN PUT FILE (PI) EDIT
(TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' ALL VOLUMES MOUNTED ',
*, (F(7), A(6), F(6), A(20), SKIP, X(15), A(3)));
IF TRACE = 1 & STEP.OP_REQUESTS ^= 0
THEN PUT FILE (PI) EDIT
(TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' DATA SETS ALLOCATED ',
*, (F(7), A(6), F(6), A(20), SKIP, X(15), A(3)));
RETURN;

STATISTIC ABEND DATA SET: ENTRY(I);
/* JOB CANCELLED BY OPERATOR, DATA SET(S) NOT AVAILABLE */

ABEND_D_SET(I) = ABEND_D_SET(I) + 1;
IF TRACE = 1
THEN PUT FILE (PI) EDIT
(TIME.SYSTEM, ' * JOB', JOB.NUMBER, ' CANCELLED BY OPERATOR ',
*, (F(7), A(6), F(6), A(22), SKIP, X(15), A(31)));
RETURN;

```

SIM11510
SIM11520
SIM11530
SIM11540
SIM11550
SIM11560
SIM11570
SIM11580
SIM11590
SIM11600
SIM11610
SIM11620
SIM11630
SIM11640
SIM11650
SIM11660
SIM11670
SIM11680
SIM11690
SIM11700
SIM11710
SIM11720
SIM11730
SIM11740
SIM11750
SIM11760
SIM11770
SIM11780
SIM11790
SIM11800
SIM11810
SIM11820
SIM11830
SIM11840
SIM11850
SIM11860
SIM11870
SIM11880
SIM11890
SIM11900
SIM11910
SIM11920
SIM11930
SIM11940
SIM11950
SIM11960
SIM11970
SIM11980
SIM11990
SIM12000
SIM12010

SIM12030
SIM12040
SIM12050
SIM12060
SIM12070
SIM12080
SIM12090
SIM12100
SIM12110
SIM12120
SIM12130
SIM12140
SIM12150
SIM12160
SIM12170
SIM12180
SIM12190
SIM12200
SIM12210
SIM12220
SIM12230

```

STATISTIC END WS: ENTRY(I);
/* UPDATE TOTAL TIME WAITING FOR D.A.SPACE */
WAIT = TIME.SYSTEM - TIME.LC(I);
MAX_WS(I) = MAX (MAX_WS(I), WAIT);
TIME_WS(I) = TIME_WS(I) + WAIT;
RETURN;

```

```

STATISTIC ABEND D A SPACE: ENTRY(I);
/* JOB ABENDED BECAUSE OF LACK OF D.A.SPACE */
ABEND_SPACE(I) = ABEND_SPACE(I) + 1;
IF TRACE = 1
  THEN PUT FILE (P1) EDIT
  (TIME.SYSTEM, ' * JOB, JOB.NUMBER, ' ABEND: NO SPACE',
  ' * '); (F(7), A(6), F(6), A(20), SKIP, X(15), A(3));
RETURN;
END STATISTIC;

```



```

READER:  PROCEDURE;
/* SIMULATE THE READER/INTERPRETER FUNCTIONS */
JOB_PTR = ADDR(INPUT(JOB_COUNT));
IF IN_SPOOL >= JOB_CARDS
THEN /* ENOUGH INPUT SPOOL SPACE, READ JCB */
DO;
  IN(JOB_CLASS) = JOB_IN(JOB_CLASS) + 1;
  JOB_ARRIVAL   = TIME.SYSTEM;
  IN_SPOOL      = IN_SPOOL - JCB.CARDS;
  CALL ENQUEUE;
  JOB_COUNT = JCB_COUNT + 1;
  IF JOB_COUNT <= MAX_JOBS
  THEN /* MORE JOBS TO READ */
  DO;
    JOB_PTR = ADDR(INPUT(JOB_COUNT));
    TIME.READER = JOB_DEL_ARRIVAL + TIME.READER;
  END; /* ALL JOBS READ IN */
ELSE;
  END;
ELSE /* NOT ENOUGH INPUT SPOOL SPACE */
/* WAIT UNTIL NEXT JCB TERMINATES */
TIME.READER = NEXT_EOJ;
END READER;

```

```

SIM12250
SIM12260
SIM12270
SIM12280
SIM12290
SIM12300
SIM12310
SIM12320
SIM12330
SIM12340
SIM12350
SIM12360
SIM12370
SIM12380
SIM12390
SIM12400
SIM12410
SIM12420
SIM12430
SIM12440
SIM12450
SIM12460
SIM12470
SIM12480
SIM12490

```

```

INITIATOR. PROCEDURE(I);
  DCL ACTIVE FIXED BIN,
  I FIXED BIN,
  N FIXED BIN,
  STATE(8) LABEL INIT
  (WAIT FOR WORK,
  REGION MANAGEMENT,
  DATA SET ALLOCATION,
  STEP_TERMINATION,
  JCB_SELECTION,
  DEVICE_ALLOCATION,
  D_A_SPACE_ALLOCATION,
  JCB_TERMINATION);

  JOB_PTR = ADDR( QUEUE(JOB_INDEX(I)));;
  STEP_PTR = ADDR( QUEUE(STEP_INDEX(I)) );;
  GOTC STATE(STATUS(I));

WAIT_FCR_WORK:
  CALL GET_MAIN(I,MIN_CORE);
  IF CORE_HIGH(I) = 0
  THEN /* NOT ENOUGH CORE */
  DO;
    TIME_INITIATOR(I) = NEXT_EOJ;
    RETURN;
  END;
  CALL DEQUEUE(I);
  IF JOB_INDEX(I) = 0
  THEN /* NO JOBS TO SERVE */
  DO;
    TIME_INITIATOR(I) = TIME.READER;
    CALL FREE_MAIN(I);
    RETURN;
  END;
  /* JOB FOUND */

  JOB_PTR = ADDR(QUEUE(JOB_INDEX(I)));
  CALL STATISTIC_START_JOB(I);
  CALL STATISTIC_END_JOB(I);
  STOP_SWITCH(I) = 1;
  TIME_LC(I) = TIME.SYSTEM;
  STATUS(I) = 3;
  RETURN;

```

```

SIM12510
SIM12520
SIM12530
SIM12540
SIM12550
SIM12560
SIM12570
SIM12580
SIM12590
SIM12600
SIM12610
SIM12620
SIM12630
SIM12640
SIM12650
SIM12660
SIM12670
SIM12680
SIM12690
SIM12700
SIM12710
SIM12720
SIM12730
SIM12740
SIM12750
SIM12760
SIM12770
SIM12780
SIM12790
SIM12800
SIM12810
SIM12820
SIM12830
SIM12840
SIM12850
SIM12860
SIM12870
SIM12880
SIM12890
SIM12900
SIM12910
SIM12920
SIM12930
SIM12940

```

```

SIM12960
SIM12970
SIM12980
SIM12990
SIM13000
SIM13010
SIM13020
SIM13030
SIM13040
SIM13050
SIM13060
SIM13070
SIM13080
SIM13090
SIM13100
SIM13110
SIM13120
SIM13130
SIM13140
SIM13150
SIM13160
SIM13170
SIM13180
SIM13190
SIM13200
SIM13210
SIM13220
SIM13230
SIM13240
SIM13250

```

```

JCB_SELECTION:
    CALL GET_MAIN(I, MIN_CORE);
    IF CORE_HIGH(I) = 0
    THEN /* NOT ENOUGH CORE */
        DO;
            TIME_INITIATOR(I) = NEXT_EOJ;
            RETURN;
        END;
    CALL STATISTIC_END_WM(I);
    TIME_LC(I) = TIME.SYSTEM;
    CALL_DEQUEUE(I);
    IF JOB_INDEX(I) = 0
    THEN /* NO JOBS TO SERVE */
        DO;
            CALL STATISTIC_BEGIN_WM(I);
            TIME_INITIATOR(I) = TIME.READER;
            STATUS(I) = 1;
            STOP_SWITCH(I) = 0;
            CALL_FREE_MAIN(I);
            RETURN;
        END;
    /* JOB FOUND */
    JOB_PTR = ADDR(QUEUE(JOB_INDEX(I)));
    STEP_PTR = ADDR(QUEUE(STEP_INDEX(I)));
    CALL_STATISTIC_START_JOB(I);
    STOP_SWITCH(I) = 1;
    STATUS(I) = 3;

```

```

REGION_MANAGEMENT:
/* FREE OLD REGION, GET NEW REGION */

CALL FREE MAIN(I);
CALL GET MAIN(I,STEP.CORE_SIZE);
IF CORE_HIGH(I) = 0
THEN /* CORE NOT AVAILABLE */
  DO;
  TIME_INITIATOR(I) = NEXT_EOJ;
  RETURN;
END;

/* NEW REGION ALLOCATED */

CALL STATISTIC-END WM(I);
TIME_LC(I) = TIME.SYSTEM;
STATUS(I) = 4;

DEVICE_ALLOCATION:
/* ALLOCATE REQUESTED DEVICES */

IF CP_ANSWER(I) < 0
THEN /* JOB CANCELLED BY OPERATOR */
  DO;
  CALL STATISTIC-END WD(I);
  CALL STATISTIC-ABEND-DEVICE(I);
  TIME_LC(I) = TIME.SYSTEM;
  STATUS(I) = 8;
  RETURN;
END;

SYSTEM.DISKS = SYSTEM.DISKS - STEP.DISKS;
IN.DISKS(I) = IN.DISKS(I) + STEP.DISKS;
SYSTEM.TAPES = SYSTEM.TAPES - STEP.TAPES;
IN.TAPES(I) = IN.TAPES(I) + STEP.TAPES;
STEP.DISKS = 0;
STEP.TAPES = 0;

```

```

SIM13270
SIM13280
SIM13290
SIM13300
SIM13310
SIM13320
SIM13330
SIM13340
SIM13350
SIM13360
SIM13370
SIM13380
SIM13390
SIM13400
SIM13410
SIM13420
SIM13430
SIM13440
SIM13450
SIM13460
SIM13470
SIM13480
SIM13490
SIM13500
SIM13510
SIM13520
SIM13530
SIM13540
SIM13550
SIM13560
SIM13570
SIM13580
SIM13590
SIM13600
SIM13610
SIM13620
SIM13630
SIM13640

```



```

IF SYSTEM.TAPES >= 0 & SYSTEM.DISKS >= C
  THEN /* ALL DEVICES AVAILABLE */
  DO;
  CALL STATISTIC_END_WD(I);
  STEP.DISKS = IN.DISKS(I);
  STEP.TAPES = IN.TAPES(I);
  TIME.LC(I) = TIME.SYSTEM;
  STATUS(I) = 5;
  GOTO DATA_SET_ALLOCATION;
END;

/* NOT ALL DEVICES AVAILABLE */
IF SYSTEM.DISKS < 0
  THEN
  DO;
  STEP.DISKS = -SYSTEM.DISKS;
  IN.DISKS(I) = IN.DISKS(I) - STEP.DISKS;
  SYSTEM.DISKS = 0;
  END;
IF SYSTEM.TAPES < 0
  THEN
  DO;
  STEP.TAPES = -SYSTEM.TAPES;
  IN.TAPES(I) = IN.TAPES(I) - STEP.TAPES;
  SYSTEM.TAPES = 0;
  END;
IF OP_ANSWER(I) = 0
  THEN /* GET OPERATOR RESPONSE */
  DO;
  CALL STATISTIC_BEGIN_WD(I);
  TIME.INITIATOR(I) = TIME.INITIATOR(I) + OP_TIME(I);
  END;
ELSE TIME.INITIATOR(I) = NEXT_EOJ;
RETURN;

```

```

SIM13660
SIM13670
SIM13680
SIM13690
SIM13700
SIM13710
SIM13720
SIM13730
SIM13740
SIM13750
SIM13760
SIM13770
SIM13780
SIM13790
SIM13800
SIM13810
SIM13820
SIM13830
SIM13840
SIM13850
SIM13860
SIM13870
SIM13880
SIM13890
SIM13900
SIM13910
SIM13920
SIM13930
SIM13940
SIM13950
SIM13960
SIM13970
SIM13980
SIM13990

```

```

DATA_SET_ALLOCATION:
7* ALLOCATE REQUESTED DATA SETS */
IF OP_ANSWER(I) < 0
  THEN
    /* JOB CANCELLED BY OPERATOR */
    DG;
    CALL STATISTIC_END WV(I);
    CALL STATISTIC-ABEND_DATA_SET(I);
    TIME_LC(I) = TIME.SYSTEM;
    STATUS(I) = 8;
    RETURN;
  END;
/* ALLOCATE DATA SETS */
IF STEP_TAPES ^= 0 | STEP_DISKS ^= 0 | STEP.OP_REQUESTS ^= 0
  THEN /* SIMULATE OPERATOR RESPONSE TIME */
    DG;
    CALL STATISTIC_BEGIN WV(I);
    TIME_INITIATOR(I) = TIME_INITIATOR(I) + OP_TIME(I);
    STEP.OP_REQUESTS = 0;
    STEP.DISKS = 0;
    STEP.TAPES = 0;
    RETURN;
  END;
/* ALL DATA SETS ALLOCATED */
CALL STATISTIC_END WV(I);
TIME_LC(I) = TIME.SYSTEM;
STATUS(I) = 6;
*/

```

```

SIM14010
SIM14020
SIM14030
SIM14040
SIM14050
SIM14060
SIM14070
SIM14080
SIM14090
SIM14100
SIM14110
SIM14120
SIM14130
SIM14140
SIM14150
SIM14160
SIM14170
SIM14180
SIM14190
SIM14200
SIM14210
SIM14220
SIM14230
SIM14240
SIM14250
SIM14260
SIM14270
SIM14280
SIM14290
SIM14300
SIM14310

```

```

D_A_SPACE ALLOCATION:
/* ALLOCATE SPACE ON DIRECT ACCESS DEVICES */
SYSTEM.D_A_SPACE= SYSTEM.D_A_SPACE - STEP.D_A_SPACE;
IN.D_A_SPACE(I)= IN.D_A_SPACE(I) + STEP.D_A_SPACE;
STEP.D_A_SPACE = 0;
IF SYSTEM.D_A_SPACE >= 0
THEN /* D_A_SPACE AVAILABLE */
DO;
CALL STATISTIC_END_WS(I);
TIME.INITIATOR(I)=TIME.INITIATOR(I) + STEP.RUN_TIME;
TIME.LC(I) = TIME.SYSTEM;
STATUS(I) = 7;
RETURN;
END;
/* NOT ENOUGH D_A_SPACE AVAILABLE */
/* CHECK IF OTHER JOB IS ACTIVE */
STEP.D_A_SPACE = -SYSTEM.D_A_SPACE;
IN.D_A_SPACE(I) = IN.D_A_SPACE(I) - STEP.D_A_SPACE;
SYSTEM.D_A_SPACE = 0;
ACTIVE = 0;
DO A=1 TO 15;
IF STATUS(N) = 7 | STATUS(N) = 8 THEN ACTIVE = 1;
END;
IF ACTIVE /= 0
THEN /* WAIT UNTIL NEXT JOB TERMINATES */
DO;
TIME.INITIATOR(I) = NEXT_EOJ;
RETURN;
END;
/* NO OTHER JOB ACTIVE. ABEND THIS JOB */
CALL STATISTIC_END_WS(I);
CALL STATISTIC_ABEND_D_A_SPACE(I);
TIME.LC(I) = TIME.SYSTEM;
STATUS(I) = 8;
RETURN;

```

SIM14330
 SIM14340
 SIM14350
 SIM14360
 SIM14370
 SIM14380
 SIM14390
 SIM14400
 SIM14410
 SIM14420
 SIM14430
 SIM14440
 SIM14450
 SIM14460
 SIM14470
 SIM14480
 SIM14490
 SIM14500
 SIM14510
 SIM14520
 SIM14530
 SIM14540
 SIM14550
 SIM14560
 SIM14570
 SIM14580
 SIM14590
 SIM14600
 SIM14610
 SIM14620
 SIM14630
 SIM14640
 SIM14650
 SIM14660
 SIM14670
 SIM14680
 SIM14690
 SIM14700
 SIM14710
 SIM14720
 SIM14730

```

STEP_TERMINATION:
/* TERMINATE STEPS */
CALL STATISTIC END STEP(I);
TIME_LC(I) = TIME.SYSTEM;
STEP_INDEX(I) = STEP_LINK(STEP_INDEX(I));
IF STEP_INDEX(I) = 0
THEN /* NO MORE STEPS */
DO;
STATUS(I) = 8;
RETURN;
END;

/* MORE STEPS */
SYSTEM.DISKS = SYSTEM.DISKS + IN.DISKS(I);
SYSTEM.TAPES = SYSTEM.TAPES + IN.TAPES(I);
IN.TAPES(I) = 0;
IN.DISKS(I) = 0;
STATUS(I) = 3;
RETURN;

```

```

SIM14750
SIM14760
SIM14770
SIM14780
SIM14790
SIM14800
SIM14810
SIM14820
SIM14830
SIM14840
SIM14850
SIM14860
SIM14870
SIM14880
SIM14890
SIM14900
SIM14910
SIM14920
SIM14930
SIM14940
SIM14950

```


SIM14970
SIM14980
SIM14990
SIM15000
SIM15010
SIM15020
SIM15030
SIM15040
SIM15050
SIM15060
SIM15070
SIM15080
SIM15090
SIM15100
SIM15110
SIM15120
SIM15130
SIM15140
SIM15150
SIM15160
SIM15170
SIM15180
SIM15190
SIM15200

```
JCB_TERMINATION:
/* TERMINATE JCBS */
SYSTEM.D.A.SPACE = SYSTEM.D.A.SPACE + IN.D.A.SPACE(I);
SYSTEM.IN.SPOOL = SYSTEM.IN.SPOOL + JOB.CARDS;
SYSTEM.TAPES = SYSTEM.TAPES + IN.TAPES(I);
SYSTEM.DISKS = SYSTEM.DISKS + IN.DISKS(I);
CALL FREE-MAIN(I);
CALL FREE-QUEUE(I);
RESOURCES(I) = 0;
CALL STATISTIC_END_JOB(I);
STATUS(I) = 2;
STOP_SWITCH(I) = 1;
TIME_SWITCH(I) = TIME-SYSTEM;
IF TIME-READER = NEXT_EOJ THEN TIME.READER = TIME.SYSTEM;
DO N=1 TO 15;
  IF TIME-INITIATOR(N) = NEXT_EOJ
  THEN TIME-INITIATOR(N) = TIME.SYSTEM;
END;
RETURN;

END INITIATOR;
```

```

TIMER: PRCCEDURE;
/* MAINTAIN THE TIME TABLE */
DCL I FIXED BIN;
IF TIME.SYSTEM = TIME.READER
THEN CALL READER;
DO I=1 TO 15;
IF TIME.SYSTEM = TIME.INITIATOR(I) & STOP_SWITCH(I) > -2
THEN
DO;
CALL INITIATOR(I);
END;
/* UPDATE THE SYSTEM TIME */
TIME.SYSTEM = TIME.READER;
DO I=1 TO 15;
IF TIME.SYSTEM > TIME.INITIATOR(I) & STOP_SWITCH(I) > -2
THEN TIME.SYSTEM = TIME.INITIATOR(I);
END;
END TIMER;

```

```

SIM15220
SIM15230
SIM15240
SIM15250
SIM15260
SIM15270
SIM15280
SIM15290
SIM15300
SIM15310
SIM15320
SIM15330
SIM15340
SIM15350
SIM15360
SIM15370
SIM15380
SIM15390
SIM15400
SIM15410
SIM15420
SIM15430
SIM15440
SIM15450

```

SIM15470
SIM15480
SIM15490
SIM15520
SIM15530
SIM15540
SIM15550
SIM15560
SIM15570
SIM15580
SIM15590
SIM15600
SIM15610
SIM15620
SIM15630
SIM15640
SIM15650
SIM15660
SIM15670
SIM15680
SIM15690
SIM15700
SIM15710
SIM15720
SIM15730
SIM15740
SIM15750

```

SUPERVISOR:
/* BODY OF SIMULATION PROGRAM */
ON ENDFILE (SYSIN) BEGIN;
  CLOSE FILE (SYSIN);
END;
ON ENCPAGE (P1) BEGIN;
  PUT FILE (P1) PAGE;
  PUT FILE (P1) LINE(9);
END;
CALL INITIALIZE_1;
DO WHILE (RESTART ^= 0);
  CALL INITIALIZE_2;
  CALL MODIFY;
  DO WHILE (TIME.SYSTEM < TIME.STCP
    & JOB_COUNT <= MAX_JOBS);
    CALL TIMER;
  END;
  TIME.SYSTEM = NEXT_EOJ THEN TIME.SYSTEM = TIME.STOP;
  IF TIME.SYSTEM = NEXT_EOJ THEN TIME.SYSTEM = TIME.STOP;
  CALL WRITE_STATISTICS;
  CALL SET_RESTART;
END;
END SIM;

```

STA00010
STA00015
STA00020
STA00040
STA00050
STA00060
STA00070
STA00080
STA00090
STA00110
STA00120
STA00130
STA00140
STA00150
STA00160
STA00170
STA00180
STA00190
STA00200
STA00210
STA00220
STA00230
STA00240
STA00250
STA00260
STA00270
STA00280
STA00290
STA00300
STA00310
STA00320
STA00330
STA00340
STA00350
STA00360
STA00370
STA00380
STA00390
STA00400
STA00410
STA00420
STA00430
STA00440
STA00450

```

STA:
PROCEDURE OPTIONS(MAIN);
/* EVALUATE AND PRINT STATISTICS */
DCL CLASS CHAR(15) INIT ('ABCDEFGHIJKLMNO');
DCL SUM(7) BIN FLOAT;
DCL IN(7) BIN FIXED;
DCL T(7) BIN FIXED;
DCL
1 SUM LC(15) FIXED BIN(31);
2 TIME-SNAP FIXED BIN(31);
3 MAXIT(15);
4 JOB-WAITM
5 MAX-SJ
6 MAX-WD
7 MAX-WS
8 MAX-WV
9 MAX-WW
10 NO(15);
11 TIME-SJ
12 TIME-WM
13 TIME-WD
14 TIME-WV
15 TIME-WS
16 TIME-WWT
17 JOB-WAITS
18 JOB-START(15)
19 IN-QUEUE
20 ABEND-DEVICE
21 ABEND-D-SET
22 ABEND-SPACE
23 CP-ANSWER
24 CP-TIME
25 JOB-IN
26 CLASSES(15)
DCL
1 TOTAL;
2 CURRENT_SNAP
3 FIRST_SNAP
4 M(15)
5 NO(15)

```



```

DCL  STAT FILE RECORD SEQUENTIAL INPUT
    ENVIRONMENT (F(1744) CONSECUTIVE);
DCL  F1 FILE PRINT;
ON  ENDFILE (STAT) BEGIN;
    GOTO SUMMARY;
END;
ON  ENDFILE (SYSIN) BEGIN;
    CLCSE FILE (SYSIN);
END;
ON  ENDPAGE (F1) BEGIN;
    PUT FILE (F1) PAGE LINE (9);
END;
OPEN FILE (F1) PAGESIZE (80);

```

```

STA00490
STA00500
STA00510
STA00520
STA00530
STA00540
STA00550
STA00560
STA00570
STA00580
STA00590
STA00600
STA00610
STA00620
STA00630
STA00640
STA00650
STA00660
STA00670
STA00680
STA00690
STA00700

```

```

FCRM1: PROCEDURE;
/* PRINT STATISTICS FORM1 */

PUT FILE (F1) PAGE LINE(9) EDIT
('INITIATORS', (NUMBER AND ASSOCIATED JCB CLASSES)', ' ')
(X(45), A(10)), SKIP, X(33), A(35), SKIP(2), X(23), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (IN(I)) (X(5), F(2));
END;

PUT FILE (F1) SKIP EDIT (' ') (X(25), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (CLASSES(I)) (X(1), A(6));
END;

PUT FILE (F1) SKIP;

DO J=1 TO 15;
  PUT FILE (F1) SKIP(2) EDIT
  ('CLASS ', SUBSTR(CLASS, J, 1), '1: ')
  (X(15), A(6), A(1), X(2), A(2));

  DO I=1 TO 7 WHILE (IN(I) ^= 0);
    PUT FILE (F1) EDIT
    (S.JOB_START(I, J)) (X(3), F(4));
  END;

  PUT FILE (F1) SKIP EDIT ('2: ') (X(24), A(2));
  DO I=1 TO 7 WHILE (IN(I) ^= 0);
    TIME = S.TIME_SJ(I) + S.TIME_WM(I) + S.TIME_WD(I);
    IF TIME ^= 0
      THEN TIME = S.JOB_START(I, J) * 60 / TIME;
    PUT FILE (F1) EDIT
    (TIME) (X(2), F(5, 3));
  END;
END;

PUT FILE (F1) SKIP(3) EDIT
('1: NUMBER CF JCBS STARTED (TOTAL)',
'2: NUMBER CF JOBS STARTED (PER MIN.)')
(X(24), A(34), SKIP, X(24), A(37));

PUT FILE (F1) SKIP(5) EDIT ('TIME OF SNAPSHOT:', TIME_SNAP)
(X(15), A(17), F(6));

END FORM1;

```

STA00720
 STA00730
 STA00740
 STA00750
 STA00760
 STA00770
 STA00780
 STA00790
 STA00800
 STA00810
 STA00820
 STA00830
 STA00840
 STA00850
 STA00860
 STA00870
 STA00880
 STA00890
 STA00900
 STA00910
 STA00920
 STA00930
 STA00940
 STA00950
 STA00960
 STA00970
 STA00980
 STA00990
 STA01000
 STA01010
 STA01020
 STA01030
 STA01040
 STA01050
 STA01060
 STA01070
 STA01080
 STA01090
 STA01100
 STA01110
 STA01120
 STA01130
 STA01140
 STA01150
 STA01160
 STA01170
 STA01180

```

FCRM2:  PROCEDURE;
/* PRINT STATISTICS FORM2 */

PUT FILE (F1) PAGE LINE(9) EDIT
('INITIATORS', '(NUMBER AND ASSOCIATED JCB CLASSES)', ' ' )
(X(45), A(10), SKIP, X(33), A(35), SKIP(2), X(23), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (IN(I)) (F(7));
END;

PUT FILE (F1) SKIP EDIT (' ' ) (X(25), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (CLASSES(I)) (X(1), A(6));
END;

PUT FILE (F1) SKIP(3) EDIT ('TIME_AC 1:') (X(15), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  SUM(I) = S.TIME_SJ(IN(I)) + S.TIME_WM(IN(I))
    + S.TIME_WD(IN(I)) + S.TIME_WV(IN(I))
    + S.TIME_WS(IN(I)) + S.TIME_WW(IN(I));
  PUT FILE (F1) EDIT (SUM(I), ' ') (F(6), A(1));
END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_SJ 1:') (X(15), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.TIME_SJ(IN(I)), ' ') (F(6), A(1));
END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24), A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.MAX_SJ(IN(I)), ' ') (F(6), A(1));
END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24), A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  IF SUM(I) = 0 THEN TIME = 0;
  ELSE TIME = S.TIME_SJ(IN(I)) * 100 / SUM(I);
  PUT FILE (F1) EDIT (TIME, ' ') (F(6), A(1));
END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WM 1:') (X(15), A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.TIME_WM(IN(I)), ' ') (F(6), A(1));
END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24), A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.MAX_WM(IN(I)), ' ') (F(6), A(1));
END;

```

STA01200
 STA01210
 STA01220
 STA01230
 STA01240
 STA01250
 STA01260
 STA01270
 STA01280
 STA01290
 STA01291
 STA01292
 STA01293
 STA01294
 STA01295
 STA01300
 STA01310
 STA01320
 STA01330
 STA01340
 STA01350
 STA01360
 STA01370
 STA01380
 STA01390
 STA01400
 STA01410
 STA01420
 STA01430
 STA01440
 STA01450
 STA01460
 STA01470
 STA01480
 STA01490
 STA01500
 STA01510
 STA01520
 STA01530
 STA01540
 STA01550
 STA01560
 STA01570
 STA01580
 STA01590
 STA01600
 STA01610
 STA01620
 STA01630

```

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  IF SUM(I) = 0 THEN TIME = 0;
  ELSE TIME = S.TIME_WM(IN(I)) * 100 / SUM(I);
  PUT FILE (F1) EDIT-(TIME,' ') (F(6,1),A(1));
END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WD 1:') (X(15),A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.TIME_WD(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.MAX_WD(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  IF SUM(I) = 0 THEN TIME = 0;
  ELSE TIME = S.TIME_WD(IN(I)) * 100 / SUM(I);
  PUT FILE (F1) EDIT-(TIME,' ') (F(6,1),A(1));
END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WV 1:') (X(15),A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.TIME_WV(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.MAX_WV(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  IF SUM(I) = 0 THEN TIME = 0;
  ELSE TIME = S.TIME_WV(IN(I)) * 100 / SUM(I);
  PUT FILE (F1) EDIT-(TIME,' ') (F(6,1),A(1));
END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WS 1:') (X(15),A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.TIME_WS(IN(I)), ' ') (F(6),A(1));
END;

```

STA01650
 STA01660
 STA01670
 STA01680
 STA01690
 STA01700
 STA01710
 STA01720
 STA01730
 STA01740
 STA01750
 STA01760
 STA01770
 STA01780
 STA01790
 STA01800
 STA01820
 STA01830
 STA01840
 STA01850
 STA01860
 STA01870
 STA01880
 STA01890
 STA01900
 STA01910
 STA01920
 STA01930
 STA01940
 STA01950
 STA01960
 STA01970
 STA01980
 STA01990
 STA02000
 STA02010
 STA02020
 STA02030
 STA02040
 STA02050
 STA02060
 STA02070
 STA02080
 STA02090
 STA02100


```

PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.MAX_WS(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  IF SUM(I) = 0 THEN TIME = 0;
  ELSE TIME = S.TIME_WS(IN(I)) * 100 / SUM(I);
  PUT FILE (F1) EDIT (TIME, ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WW 1:') (X(15),A(1));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.TIME_WW(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  PUT FILE (F1) EDIT (S.MAX_WW(IN(I)), ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ^= 0);
  IF SUM(I) = 0 THEN TIME = 0;
  ELSE TIME = S.TIME_WW(IN(I)) * 100 / SUM(I);
  PUT FILE (F1) EDIT (TIME, ' ') (F(6),A(1));
END;

PUT FILE (F1) SKIP(3) EDIT
('TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES),
'TIME_SJ: TIME SERVING JOBS (=ELAPSED JOB RUN TIME),
'TIME_WM: TIME WAITING FOR MAIN MEMORY
'TIME_WD: TIME WAITING FOR DEVICE(S)
'TIME_WV: TIME WAITING FOR VOLUME(S) TC BE MOUNTED
'TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
'TIME_WW: TIME WAITING FOR WORK
'TIME_TCTAL: TOTAL TIME IN SEC.
'1: MAXIMUM TIME IN SEC.
'2: TIME IN % OF ACTIVE TIME
'3: TIME IN % OF ACTIVE TIME
(X(15),A(59)), SKIP, X(15),A(51)), SKIP, X(15),A(51)),
SKIP, X(15),A(51)), SKIP, X(15),A(51)), SKIP, X(15),A(51)),
SKIP, X(15),A(51)), SKIP, X(15),A(51)), SKIP, X(15),A(51)),
PUT FILE (F1) SKIP(5) EDIT
('TIME OF SNAPSHOT:', TIME_SNAP) (X(15),A(17),F(6));
END FORM2;

```

STA02120
 STA02130
 STA02140
 STA02150
 STA02160
 STA02170
 STA02180
 STA02190
 STA02200
 STA02210
 STA02220
 STA02230
 STA02240
 STA02250
 STA02260
 STA02270
 STA02280
 STA02290
 STA02300
 STA02310
 STA02320
 STA02330
 STA02340
 STA02350
 STA02360
 STA02370
 STA02380
 STA02390
 STA02400
 STA02420
 STA02430
 STA02440
 STA02450
 STA02460
 STA02470
 STA02480
 STA02490
 STA02500
 STA02510
 STA02520
 STA02530
 STA02540
 STA02550
 STA02570
 STA02580
 STA02590
 STA02600

```

FCRM3: PROCEDURE;
/* PRINT STATISTICS FORM3 */

PUT FILE (F1) PAGE LINE(9) EDIT
('J.A.#',J.S.#,'J.S.#','MAX.W','MEAN W','DEV.W');
(X(23),A(6),X(2),A(6),X(2),A(6),X(2),A(6),X(2),A(6));
DO J=1 TO 15;
  JOBS_AVAIL = S.JOB_IN(J) + S.IN_QUEUE(J);
  START = 0;
  DO I=1 TO 15;
    START = START + S.JOB_START(I,J);
  END;
  PUT FILE (F1) SKIP(2) EDIT
  ('CLASS',SUBSTR(CLASS,J,1),JOBS_AVAIL,START)
  (X(15),A(6),A(1),F(6),F(8));
  IF START = 0 THEN PUT FILE (F1) EDIT
  ('---',---,---) (X(6),A(2),X(6),A(2),X(6),A(2));
  ELSE DO;
    AVERAGE = S.JOB_WAIT(J) / START;
    IF S.JOB_IN(J) = 0 THEN PERCENT = 0;
    ELSE PERCENT = START * 100 / JOBS_AVAIL;
    PUT FILE (F1) EDIT
    (PERCENT,S.JOB_WAITM(J),AVERAGE) (F(8,1),2F(8));
  END;
  IF START < 2 THEN PUT FILE (F1) EDIT ('--') (X(6),A(2));
  ELSE DO;
    VARIANCE = (S.JOB_WAITS(J) - START * AVERAGE ** 2)
    / (START - 1);
    DEVIATION = SQRT(VARIANCE);
    PUT FILE (F1) EDIT (DEVIATION) (F(8));
  END;
END;

PUT FILE (F1) SKIP(3) EDIT
('J.A.# = JOBS_AVAILABLE (TOTAL NUMBER)
J.S.# = JOBS STARTED (TOTAL NUMBER)
J.S.# = JOBS STARTED (IN % OF JOBS AVAILABLE)
MAX.W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
DEV. W = STANDARD DEVIATION OF WAITING TIME
(X(15),A(58),SKIP,X(15),A(58),SKIP,X(15)
A(58),SKIP,X(15),A(58),SKIP,X(15),A(58));
PUT FILE (F1) SKIP(5) EDIT ('TIME OF SNAPSHOT:',TIME_SNAP)
(X(15),A(17),F(6));

END FORM3;

```

```

STA02620
STA0264J
STA02650
STA02660
STA0267J
STA02690
STA02700
STA02710
STA02720
STA0273J
STA02740
STA02760
STA0277J
STA02780
STA02800
STA02810
STA02820
STA02830
STA02840
STA02850
STA02860
STA02870
STA02880
STA02900
STA02910
STA02920
STA02930
STA02940
STA02950
STA02960
STA02970
STA02990
STA03000
STA03010
STA03020
STA03030
STA0304J
STA03050
STA03060
STA03070
STA03090
STA0310J
STA03120

```

```

PREPARE: PROCEDURE;
/* PREPARE SUMMARY */
CURRENT_SNAP = TIME_SNAP;
TOTAL = TOTAL + S, BY NAME;
TOTAL.IN_QUEUE = 0;
N = N + 1;
IF N=1 THEN FIRST_SNAP = TIME_SNAP;
DO I=1 TO 15;
M.JOB_WAITM(I) = MAX(M.JOB_WAITM(I), S.JOB_WAITM(I));
END;

DO I=1 TO 7 WHILE (IN(I) <= 0);
M.MAX_SJ(IN(I)) = MAX (M.MAX_SJ(IN(I)), S.MAX_SJ(IN(I)));
M.MAX_WD(IN(I)) = MAX (M.MAX_WD(IN(I)), S.MAX_WD(IN(I)));
M.MAX_WM(IN(I)) = MAX (M.MAX_WM(IN(I)), S.MAX_WM(IN(I)));
M.MAX_WS(IN(I)) = MAX (M.MAX_WS(IN(I)), S.MAX_WS(IN(I)));
M.MAX_WV(IN(I)) = MAX (M.MAX_WV(IN(I)), S.MAX_WV(IN(I)));
M.MAX_WW(IN(I)) = MAX (M.MAX_WW(IN(I)), S.MAX_WW(IN(I)));
END;

END PREPARE;

```

```

STA03140
STA03150
STA03160
STA03170
STA03180
STA03190
STA03200
STA03210
STA03220
STA03230
STA03240
STA03250
STA03260
STA03270
STA03280
STA03290
STA03300
STA03310
STA03320
STA03330
STA03340
STA03350

```

STA03370
 STA03380
 STA03390
 STA03400
 STA03410
 STA03420
 STA03430
 STA03440
 STA03450
 STA03460
 STA03470
 STA03480
 STA03490
 STA03500
 STA03510
 STA03520
 STA03530
 STA03540
 STA03550
 STA03560
 STA03570
 STA03580
 STA03590
 STA03600
 STA03610
 STA03620
 STA03630
 STA03640
 STA03650
 STA03660
 STA03670
 STA03680

```

TOTAL = 0;
N = 0;
PUT LIST ('ENTER INITIATORS:');
PUT SKIP;
GET LIST (IN(1), IN(2), IN(3), IN(4), IN(5), IN(6), IN(7));

LCCP:
  READ FILE (STAT) INTO (S);
  PUT LIST ('ENTER STATISTICS TYPES:');
  PUT SKIP;
  GET LIST (T(1), T(2), T(3), T(4));
  IF T(4) = 1 THEN CALL PREPARE;
  IF T(1) = 1 THEN CALL FORM1;
  IF T(2) = 1 THEN CALL FORM2;
  IF T(3) = 1 THEN CALL FORM3;
  GCTC LCCP;

SUMMARY:
  PUT SKIP LIST ('ENTER SUMMARY TYPES:');
  PUT SKIP;
  GET LIST (T(1), T(2), T(3));
  TIME_SNAP = CURRENT_SNAP;
  S = TOTAL, BY NAME;
  MAXI = M;
  IF T(1) = 1 THEN CALL FORM1;
  IF T(2) = 1 THEN CALL FORM2;
  IF T(3) = 1 THEN CALL FORM3;
  END STA;
  
```


BIBLIOGRAPHY

1. Afifi, A.A. and Azen, S.P., Statistical Analysis, A Computer Oriented Approach, Academic Press, 1972.
2. Barron, D.W., Computer Operating Systems, Chapman and Hall Ltd., 1971.
3. Browne, J.C., Ian, J. and Baskett, P., "The Interaction of Multiprogramming Job Scheduling and CPU Scheduling", Proceedings, AFIPS, FJCC, vol. 41, part 1, p. 13-22, 1972.
4. Colin, A.J.T., Introduction to Operating Systems, American Elsevier Publishing Co., 1971.
5. Crowley J.J.Jr., Lt USN, and Karns, L.N., Lt USN, An Interactive Simulation of the IBM System 360 Operator's Console, M.S. Thesis, Naval Postgraduate School, Monterey, California 93940, 1976.
6. Cuttle, G. and Robinson, P.B., Executive Programs and Operating Systems, American Elsevier Publishing Co., 1970.
7. Flores, I., Computer Programming System/360, Prentice-Hall, 1971.
8. Flores, I., Job Control Language and File Definition, Prentice-Hall, 1971.
9. Flores, I., QS/MVT, Prentice-Hall, 1973.
10. Freeman, P., Software Systems Principles, Science Research Associates, 1975.

11. Hoare, C.A.R. and Perrot, R.H., Operating Systems Techniques, Academic Press, 1972.
12. IBM System/360 Operating System: Introduction, 5th ed., IBM, 1972.
13. IBM System/360 Operating System: MVT Guide, 6th ed., IBM, 1974.
14. IBM System/360 Operating System: MVT Job Management, Program Logic Manual, 10th ed., IBM, 1971.
15. IBM System/360 Operating System: MVT Supervisor, 7th ed., IBM, 1972.
16. IBM System/360 Operating System: Operator's Reference, OS Release 21.7, 4th ed., IBM, 1974.
17. Katzan, H., Jr., Computer Organization and the System/370, Van Nostrand Reinhold, 1971.
18. Katzan, H., Jr., Operating Systems, Van Nostrand Reinhold, 1973.
19. Katzan, H., Jr., Information Technology, Petrocelli Books, 1974.
20. Madnick, S.E. and Donavan, J.J., Operating Systems, McGraw-Hill Book Co., 1974.
21. Sayers, A.P. , editor, Operating Systems Survey, Auerbach Publishers, 1971.
22. User's Manual, W.R. Curch Computer Center, 2nd ed., Naval Postgraduate School, Monterey, California, 1974.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4. Professor N. F. Schneidewind, Code 52SS Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5. Professor R. J. Roland, Lt.Col., USAF, Code 52RL Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6. Professor D. J. Williams Director W. R. Church Computer Center, Code 0141 Naval Postgraduate School Monterey, California 93940	1

- | | | |
|-----|---|---|
| 7. | Mr. D. F. Norman | 1 |
| | Manager Operations Group, Code 0141 | |
| | Naval Postgraduate School | |
| | Monterey, California 93940 | |
| 8. | Marineamt -A1- | 1 |
| | 294 Wilhelmshaven | |
| | Federal Republic of Germany | |
| 9. | Dokumentationszentrale der Bundeswehr (See) | 1 |
| | 53 Bonn | |
| | Friedrich-Ebert-Allee 34 | |
| | Federal Republic of Germany | |
| 10. | BMVg - P V 13 - | 1 |
| | 53 Bonn | |
| | Postfach | |
| | Federal Republic of Germany | |
| 11. | KptLt Erik Fiegl | 1 |
| | Marinefernmeldegruppe 11 | |
| | 2392 Gluecksburg-Meierwik | |
| | Federal Republic of Germany | |